DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE VOZ Y UN SISTEMA DE DIÁLOGO BASADO EN MÁQUINAS DE ESTADO FINITO PARA EL CONTROL DE UNA PLATAFORMA MÓVIL.

Autor:
DANIEL EDUARDO ESCALANTE CORDOBA

Universidad Tecnológica de Pereira
Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación
Programa de Ingeniería Electrónica
Pereira-Risaralda
2018

DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE VOZ Y UN SISTEMA DE DIÁLOGO BASADO EN MÁQUINAS DE ESTADO FINITO PARA EL CONTROL DE UNA PLATAFORMA MÓVIL.

Autor:
DANIEL EDUARDO ESCALANTE CORDOBA

Proyecto de grado para optar por el título de Ingeniero Electrónico

Director
Julián David Echeverry Correa, PhD.

Universidad Tecnológica de Pereira
Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación
Programa de Ingeniería Electrónica
Pereira
2018

Agradecimientos

Quiero agradecer a la Universidad Tecnológica de Pereira por brindarme la oportunidad de desarrollar mis habilidades cognitivas, a mis padres por el apoyo incondicional. Agradecimiento profundo al PhD Julián David Echeverry quien me acompañó en el inicio de mi vida laboral y en la finalización de mi carrera profesional. A Gustavo Sánchez por el apoyo brindado en la carrera. Agradecer a todas las personas que me acompañaron y me permitieron adquirir nuevos conocimientos.

Tabla de contenido

Índice de figuras

Índice de Tablas

1. INTRODUCCIÓN

El ser humano desde su nacimiento busca la manera de poderse comunicar con sus iguales, de manera que pueda expresar la voluntad, el deseo, el afán, y la necesidad de que compartamos un hilo común en nuestros pensamientos [1], [2].

El habla es el medio de comunicación más empleado a nivel mundial porque es una habilidad que al poco tiempo de haber nacido gran parte de la población inicia su desarrollo (el habla), es así como el habla se convierte en una herramienta para poder comunicarse con familiares y con el mundo, siendo éste el sonido que emitimos emanando aire a través de nuestra boca y tras haber hecho vibrar las cuerdas vocales que se ubican entre la laringe y los pulmones, transportándose a través del aire y siendo captado por los oídos de los receptores [1], [2].

El agitado ritmo de vida y los consecuentes niveles de estrés han llevado a los seres humanos a idear y construir dispositivos capaces de interactuar con el hombre permitiendo simplificar tareas, e incluso ejecutarlas [3].

Una de las técnicas más comunes de interactuar con la máquina es mediante de reconocimiento de voz, dado que el hombre se puede comunicar de una manera natural con las máquinas [3].

El propósito de este trabajo tipo experimental, es permitir incursionar en el campo del reconocimiento de voz, el cual será abordado a partir del estudio del estado del arte, la generación de modelos acústicos y de lenguaje. Este proyecto consiste en el desarrollo de un sistema de reconocimiento de voz automático y un sistema de diálogo basado en máquinas de estado finito para el control de una plataforma móvil. Para el desarrollo de este proyecto se va a recurrir al software de desarrollo HTK, con el cual con el cual se podrá implementar un sistema de reconocimiento de habla “palabras aisladas”.

2. PLANTEAMIENTO DEL PROBLEMA

Desde el nacimiento de la computación digital, el reconocimiento automático de voz ha estado en un constante desarrollo. Esto ha permitido no solo mejorar la interacción entre el hombre y las máquinas, sino también disminuir las brechas de comunicación entre las personas.

Esta tecnología se ha extendido y hoy en día su uso tiene aplicaciones en sectores tan diversos como el de la seguridad, la domótica, la industria, la salud, entre otros. Esto se debe a que se ha convertido en una herramienta capaz de disminuir el margen de error a causa del factor humano, y al mismo tiempo aumentar el rendimiento del personal de manera grupal o individual [4][5][6].

En el campo de la inclusión social, la Universidad Pedagógica Nacional de Bogotá - Colombia, está desarrollando un sistema inteligente de reconocimiento de voz para la traducción del lenguaje oral a la lengua de señas colombiana con el fin de la inclusión de los estudiantes sordos a la formación en la educación superior, para que de este modo se pueda brindar apoyo y facilitar la comunicación con los docentes y compañeros de clase [7].

En el ámbito de las aplicaciones tecnológicas, en la Universidad Tecnológica de Pereira el Grupo de Investigación en Automática desde hace unos años ha venido incursionando en el campo del reconocimiento automático de voz. En este grupo se han realizado distintos trabajos, entre los que destacan el diseño de un *Sistema de reconocimiento de voz aislada utilizando Arduino Micro y Bitvoicer Server para control domótico simulado* y el *diseño e implementación de un sistema de reconocimiento de voz mediante Raspberry Pi* [3][6].

Actualmente a nivel nacional, los sistemas que explotan la interacción entre las máquinas y los usuarios son de naturaleza mecánica. Es decir, el usuario no recibe sugerencias, ni se retroalimenta de los errores pasados, convirtiendo la labor en rutina mecánica (encender-iniciar proceso-finalizar proceso-apagar) [8].

En la Universidad Tecnológica de Pereira no se ha desarrollado en el campo de sistemas de interacción persona máquina mediante interfaces de habla capaz de comunicarse de manera alámbrica con una plataforma móvil.

Por lo anterior, se diseñó un sistema automático de reconocimiento de voz el cual funciona de manera conjunta con un gestor de diálogo, para el control de la plataforma móvil de manera alámbrica.

Este trabajo buscar impactar positivamente en el área de los desarrollos actuales en la medida que es el diseño de un sistema apto para interactuar con el usuario, brindando retroalimentación al usuario, teniendo la capacidad de evitar ambigüedades propias del lenguaje [9][10][11].

3. ESTADO DEL ARTE

3.1. Sobre el reconocimiento de voz

En la Universidad de San Carlos de Guatemala, Genoveva Velásquez Ramírez, diseñó en abril de 2008 un "sistema de reconocimiento de voz en Matlab" para optar por el grado de ingeniera electrónica. está tesis presenta un sistema de reconocimiento de voz con la capacidad de procesamiento digital de señales de voz, el cual realiza un análisis gráfico en el reconocimiento, este consiste en realizar un análisis del espectro de frecuencias de la señal de voz. el sistema posee un entorno gráfico en la computadora, lo que proporciona la selección de grabación, donde se ingresa la señal de voz por medio de un micrófono a la computadora y esta es procesada por los algoritmos que nos permiten obtener los parámetros significativos de la señal de voz y estas son almacenada, para luego ser reconocidas en la base de datos que exista en ese momento. esto, por otra parte, nos proporciona un análisis gráfico de las palabras que han sido grabadas y reconocidas [12].

En la Universidad de San Buenaventura Bogotá, Colombia, Javier Alfredo Báez Dávila, diseñó en el 2006 "Un dispositivo para el reconocimiento de caracteres vocálicos, para ordenar comandos al televisor". Como resultado se obtuvo que el dispositivo puede obtener respuesta a las órdenes transmitidas por voz a voz al televisor encendido apagado entre otros. Con este modo de relación liberará al usuario del uso de la vista y de las manos, es decir de la pantalla y el control, así dejando al usuario libre de movimiento. Proyecto en el cual los resultados obtenidos por el dispositivo demuestran que el uso de esta técnica permite obtener un 80% de clasificación correcta.
Además de esto, describe la realización de un prototipo capaz de interpretar comandos vocales basados en un microcontrolador (PIC 18 F 452) como lo es el Hardware y la implementación del Software necesario [13].

En la Universidad Tecnológica de Pereira, Colombia, Edicson Alonso Alzate Vanegas, en el 2016, presentó la tesis "Desarrollo de un sistema de reconocimiento de comandos de habla aislada para el control automático de un vehículo simulado". Este trabajo permitió incursionar en el campo del reconocimiento de voz, mediante la implementación de un sistema de reconocimiento de habla, para ser más específico de "palabras aisladas", con el software de HTK. Este sistema se desarrolló con el objetivo de realizar un control simulado de un vehículo mediante una interfaz gráfica en Matlab que simule el funcionamiento del vehículo [8].

3.2. Sobre el gestor de diálogo

En la Universidad Carlos III de Madrid, España, Víctor Corrales Muñoz, en el 2010, presentó la tesis "Desarrollo de un entorno para la interacción multimodal con diferentes aplicaciones en xhtml+voice" el cual consistió en el estudio y desarrollo de una aplicación basada en la tecnología que permite la interacción multimodal entre usuario y máquina. Las tecnologías utilizadas están basadas en sistemas de diálogo orales integrados en programas tales como el navegador web Opera, y desarrollados mediante aplicaciones diseñadas mediante lenguajes de programación como XHTML+Voice [14].

En la Universidad Politécnica de Valencia, España, Francisco Torres Goterris, en el 2006, presentó la tesis, titulada "Sistemas de diálogo basados en modelos estocásticos". Está tesis presenta el diseño e implementación de los módulos de un sistema de diálogo. Este documento se centra en el estudio de la gestión de diálogo desde una aproximación estadística, aportando al desarrollo de un sistema de diálogo completo (con entrada y salida de texto, en lengua española, y para una tarea de dominio semántico restringido, la definida en el proyecto de investigación BASURDE). Este sistema está constituido por los módulos de comprensión del lenguaje natural, de gestión del diálogo y de generación de respuestas en lenguaje natural [15].

En la Universidad Nacional, Colombia, William Alfonso Arévalo Camacho, en el 2010, presentó la tesis, "Método de conversión de un diálogo controlado a un discurso en UN-LENCE", el cual procura de disminuir la ambigüedad existente en el diálogo, se desarrollaron los lenguajes controlados, que son subconjuntos del lenguaje natural. Los lenguajes controlados poseen una estructura similar al lenguaje natural, con reglas léxicas, reglas gramaticales, signos y palabras. Los lenguajes controlados tienen diversos usos. En la especificación de requisitos de software se encuentra el lenguaje controlado UN-Lencep, que permite presentar del discurso del interesado de una forma que se pueda validar, adicionalmente, presenta la información de manera concreta, sin ambigüedades y completa. Sin embargo, la información se debe obtener mediante el diálogo con el interesado, dejando en manos del analista la identificación de los elementos necesarios para la estructuración del discurso, lo que posibilita la aparición de errores. Para reducir esta problemática, se empleó la estructuración de una secuencia ordenada de preguntas y la definición de las reglas necesarias para convertir las respuestas en el discurso del interesado, expresado en UN-Lencep. Adicionalmente, en un prototipo funcional se incluyen estos elementos y se valida con la especificación de algunos proyectos que requieren el desarrollo de una aplicación de software [16].

3.3. Sobre la plataforma móvil

En la Universidad Tecnológica de Mixteca, México, Gerardo Israel Palafox Alvarado, en el 2009, presento la tesis “Diseño y construcción de un vehículo eléctrico con variador de velocidad mediante un convertidor CD-CD”. Este trabajo consistió en la construcción de un vehículo monoplaza accionado por un sistema electrónico de potencia con el cual se regula la velocidad de un motor de corriente directa (CD) [17].

En la Universidad de la Salle de Bogotá, Colombia, Juan Elías Rincón, en el 2008, presentó la tesis “Diseño y construcción de un dispositivo electrónico para la detección de obstáculos, como ayuda a personas con discapacidad visual” metodología utilizada para la elaboración de este trabajo, es estudiar los tipos de instrumentos que se utilizan, las técnicas o recomendaciones para la ayuda de personas con discapacidad visual, los tipos de sensores de proximidad más comunes, y diseñar, mediante la información obtenida, un dispositivo para la detección de obstáculos, fácil de utilizar y económico, que permita al invidente saber por medio de alarmas la distancia a la que se encuentra el obstáculo, para que este tipo de tecnologías sean asequibles a personas invidentes de bajos recursos [18].

En la Universidad Tecnológica de Pereira, Colombia, Julio Cesar Yepes Valencia y Marlon Andrey Fernández Mesa, en el 2016 presentaron la tesis “PMITO – Plataforma móvil para investigación de técnicas de edometría”. Este trabajo describe el diseño y construcción de una plataforma móvil con anillos de sensores infrarrojos y sensores ultrasónicos como dispositivos para la medición de distancia, los cuales serán posteriormente utilizados para la navegación de robots móviles en ambientes dinámicos. Se muestra una breve reseña sobre la arquitectura de robots móviles y la importancia de determinar una estructura de control para formular alternativas de navegación y construcción de mapas de entorno con la implementación de sensores que permitan al sistema conocer la distancia que ha recorrido, su ubicación, y la distancia existente entre el entorno y el sistema mismo, utilizando sistemas embebidos, en este caso tarjetas de la serie Arduino [19].

4. OBJETIVOS

Objetivo General

Desarrollar un sistema de reconocimiento de voz y un sistema de diálogo basado en máquinas de estado finito para el control de una plataforma móvil.

Objetivos Específicos

1. Generar una base de datos de comandos de voz para el entrenamiento y evaluación del sistema de reconocimiento de voz.

2. Desarrollar e implementar un sistema de reconocimiento de voz de habla aislada para el reconocimiento de comandos por voz para el control de una plataforma móvil

3. Implementar un gestor de diálogo para la interacción entre el usuario y la plataforma móvil.

4. Establecer un sistema electrónico de comunicación entre el gestor de diálogo y la plataforma móvil.

5. Evaluar el funcionamiento y rendimiento de todo el sistema.

5. MARCO TEÓRICO

A continuación, se presenta el marco teórico, donde se explica por categorías los conceptos necesarios para el desarrollo de este proyecto.

En primera instancia se explican los conceptos como el reconocimiento de voz, gestor de diálogo, el modelo acústico, y sus etapas. Seguido a esto se procede a explicar el hardware, donde se habla acerca del sistema embebido empleado y sus respectivos módulos. Por último, en la categoría de software, se procede a explicar los programas que se emplearon; Matlab para el diseño del gestor de diálogo y HTK para el sistema de reconocimiento de voz.

CONCEPTOS

5.1. MODELO ACÚSTICO

El modelo acústico es la representación matemática y estadística de los patrones que describen la voz.
En la figura 1 se presentan los pasos necesarios para el entrenamiento de los modelos, donde en primera instancia se realiza la extracción de información (base de datos de las grabaciones de voz y su respectiva parametrización), seguido a esto se extraen los coeficientes cepstrum en la frecuencia de Mel (MFCC), los cuales se organizan los cuales se organizan de forma vectorial , teniendo que cada vector se componen de 39 coeficientes, de los cuales 13 son los coeficientes MFCCs, otros 13 son componentes de velocidad, es decir, la primera derivada y los últimos 13 componentes son la aceleración, es decir la segunda derivada.

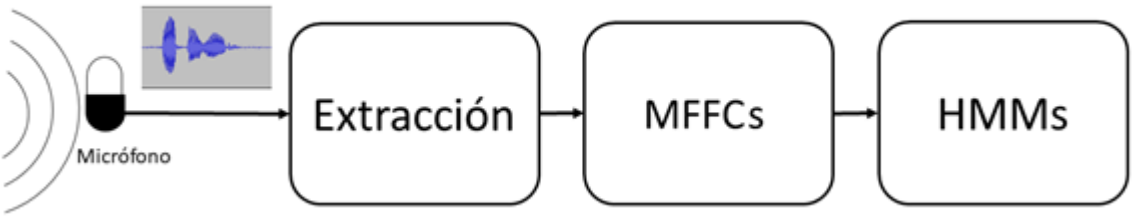

Figura 1. Modelo acústico.

A continuación, se presentan con más detalle los conceptos de MFFCs y HMMs con el objeto de conocer la importancia de estos conceptos sobre el proyecto a desarrollar.

5.2. MODELOS EN LA FRECUENCIA DE MEL (MFCC)

El análisis acústico consiste en la conversión de una señal de onda a un vector de coeficientes cepstrales en la frecuencia de MEL (*MFCC*).
Los *MFCC* son coeficientes para la representación del habla basados en la percepción auditiva, donde cabe aclarar que el oído humano resuelve el problema de frecuencias no lineales a través del espectro. Evidencia empírica sugiere que el diseño de una interfaz que cumpla las funciones de forma no lineal mejorara el desempeño. Una alternativa para no tener que trabajar con un sistema no lineal, es tratar la señal con un banco de filtros pasa bandas triangulares, a su vez espaciados en la escala MEL y con un traslape, para que de esta forma se pueda trabajar de una forma lineal [20], [4], [5], [6].
En la figura 2 se presenta el diagrama de bloques donde se puede observar una manera simple de resumir el análisis acústico para la obtención de los coeficientes *MFCC*. En la figura lo primero que se encuentra es un filtro(PE) de preénfasis, el cual busca suavizar el espectro, realzar las altas frecuencias y de este modo aumentar la relación señal a ruido. Continuó a este bloque de proceso está el ventaneo o muestreo(W) a la señal obtenida tras el preénfasis, para acotarla. Una vez que a la señal se le realizo su respectivo ventaneo, se da paso al dominio de la frecuencia mediante la FFT (Fast Fourier Transform - FFT). Seguido a esto se calculan los valores del banco de filtros distribuidos en frecuencia, según la escala MEL (MF). Finalmente, calculamos el logaritmo de dicho banco, y mediante la DCT (Discret Fourier Transform – DCT), obteniendo así el vector de los coeficientes cepstrales de las frecuencias de MEL [20].

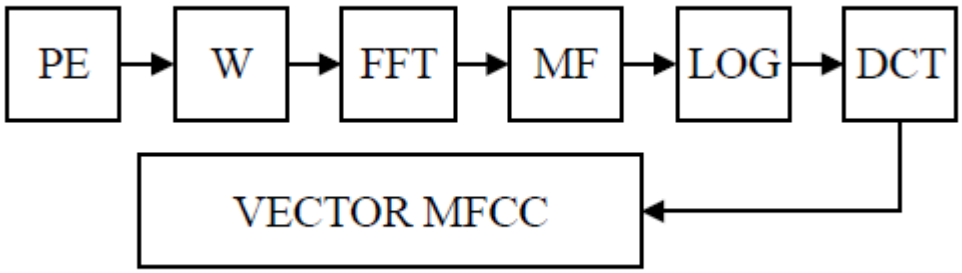

Figura 2. Esquema de obtención de los MFCC.

Para ampliar el concepto de análisis acústico, en la ecuación 1 se presenta la multiplicación que se debe realizar por cada trama de la señal con el fin de extraer los coeficientes con la transformada discreta coseno de Fourier.

$$\text{Mel(f)}= 2595 * log_{10}(1+\frac{f}{700})$$

Ecuación 1. Ecuación logarítmica escala MEL.

En la ecuación 2 se presenta cómo se obtienen nuevamente los 39 coeficientes que representan la señal de voz normalizada. Donde i y J son contadores, C son los

coeficientes, N es el número de filtros triangulares y m_J es los coeficientes a la salida del banco de filtros.

$$Ci=\sqrt{\frac{2}{N}} * \sum_{J=1}^{N} m_J * \cos(\frac{\pi}{N} * (J - 0{,}5))$$

Ecuación 2. Coeficientes de la transformada discreta de Fourier.

5.3. MODELOS OCULTOS DE MARKOV

El modelo oculto de Markov es una de las metodologías más empleadas en temas relacionados con aprendizaje de máquinas, en este caso cabe resaltar esta metodología para el procesamiento del habla y el lenguaje. Debido a que los *HMM* son un modelo de secuencia cuyo objetivo es asignar una etiqueta o una clase a cada unidad en una secuencia, y de esta manera mapear a una secuencia de etiquetas [21].
Los *HMM* son un método probabilístico de secuencia, esto quiere decir que a partir de una secuencia de unidades (palabras, letras, oraciones, entre otros) se puede calcular una probabilidad de distribución sobre secuencias posibles de etiquetas, con el objetivo de definir la mejor secuencia de etiquetas.
Para entender y definir bien los modelos ocultos de Markov es necesario conocer la cadena de Markov.

CADENAS DE MARKOV

Una cadena de Markov es un proceso estocástico que modela el comportamiento de un proceso con características aleatorias en el transcurso del tiempo, utilizando una familia de variables aleatorias parametrizadas. Un proceso aleatorio se caracteriza por presentar eventos que no siguen una regla determinada y por lo tanto no es fácil definir con certeza una secuencia de eventos previamente (a prior) en forma segura antes de que estos ocurran [21], [22].

MODELOS OCULTOS DE MARKOV (*HMM*)

Los modelos ocultos de Markov (*HMM*) constituyen una herramienta de modelamiento altamente flexible, inicialmente utilizada en el campo del reconocimiento automático del habla, que ha encontrado en los últimos años numerosas aplicaciones en áreas científico-técnicas muy diversas [3], [21], [22].

Cada uno cuenta con un modelo oculto de Markov, el cual tiene una topología determinada por:

- Numero de estados.

- Forma de funciones (Asociado a cada estado).
- Disposición de las transiciones de estados.

En la figura 3 se muestra una máquina de 6 estados finita, es decir un modelo oculto de Markov, con 4 estados activos (S2, S3, S4, S5) y dos son estados no emisores (S1, S6), esto quiere decir sin función de observación. Una vez identificado los estados, las posibles transiciones (ai) de estados y las distribuciones gaussianas (bj), se diseña la matriz de estados la cual se presenta en la figura 3, mostrando las posibles transiciones por cada estado. Los componentes de la matriz de transiciones *aij* representan la probabilidad que hay de ir de un estado i a otro estado diferente j y/o la probabilidad de estar en un estado y retornar al mismo estado [3], [21], [22].

Propiedades de la matriz de transiciones.
1. La probabilidad de cualquier estado es mayor a cero y menor a uno.
2. La probabilidad es cero si no existe conexión entre el punto de inicio y el de llegada.
3. La suma de las probabilidades de cada fila de la matriz A es igual a 1.

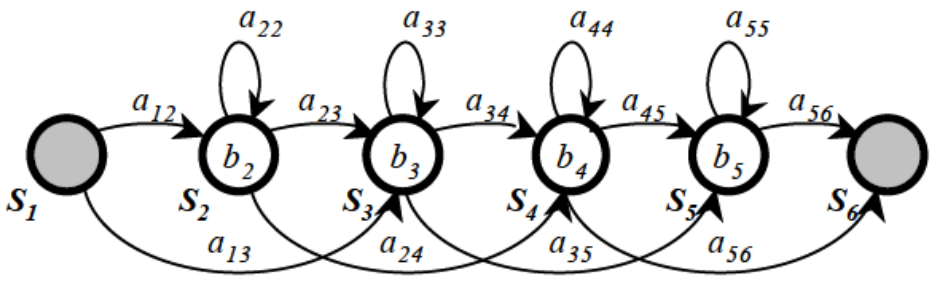

Figura 3. Máquina de estados finita [1]. En la figura se evidencia que en cualquier instante de tiempo especificado se puede considerar que el sistema está en uno de los estados disponibles y que, a intervalos regulares de tiempo ocurre una transición de un estado a otro (o al mismo estado si este dispone de una transición así mismo) conforme a las probabilidades asociadas a los arcos de transición [5.N].

Dado el diagrama de 6 estados de la figura 3, se obtiene la matriz de transiciones A, la cual se presenta en la figura 4, donde por cada estado se definen todas sus posibles transiciones.

$$A = \begin{bmatrix} 0 & a_{12} & a_{13} \\ a_{22} & a_{23} & a_{24} \\ a_{33} & a_{34} & a_{35} \\ a_{44} & a_{45} & a_{46} \\ a_{55} & a_{56} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Figura 4. Matriz de transiciones.

Para realizar la definición formal de los *HMM*, se emplea una notación compacta:

$\lambda = (A,B,\pi)$

Para indicar el conjunto de parámetros completos del modelo. Este conjunto de parámetros, por supuesto, definen una medida de probabilidad para O, por ejemplo, $P(O|\lambda)$, donde:

λ : Representa el modelo

A : Es la matriz de distribución de probabilidad de cada estado de transición

B : Es la matriz de observación de los símbolos de la distribución.

π : Representa la matriz de distribución del estado inicial.

5.4. GESTOR DE DIÁLOGO

Un gestor de diálogo es el encargado de establecer la comunicación entre el sistema de reconocimiento de voz y la interfaz hombre máquina, permitiendo de este modo que el sistema trabaje en armonía y sincronizado, es decir que la acción del sistema en general sea acorde al comando que se solicitó.

Las técnicas para la construcción de modelos de diálogo, se basan en la identificación y clasificación de los actos del habla y en la forma en que se pueden encontrar en un diálogo cualquiera [23].

La estructura de cualquier discurso consta de tres componentes: la estructura de la secuencia de las unidades del habla, la estructura de las intenciones y el estado de atención. Con estos tres componentes, es posible determinar los actos del habla de un discurso cualquiera y las diferentes relaciones que puedan poseer [24], [25].

El análisis de discursos, es el modelo que se propone para el desarrollo de este proyecto, el cual se puede interpretar como diálogos orientados a tareas específicas (diálogos en donde uno de los interlocutores busca que los otros participantes realicen alguna acción) haciendo algunas suposiciones en su modelo [26], [27].

Estas funciones se asocian con cada acto del habla y su uso puede requerir que se cumplan condiciones (pre condiciones o pos condiciones) al interior del diálogo [28]. En la figura 5 se presenta un ejemplo donde se muestran tres tipos de vehículos y accesorios de cada uno, también se presenta que accesorios tiene en común, de esta manera permite observar que los accesorios deben cumplir la condición de que sean objetos comunes en los vehículos.

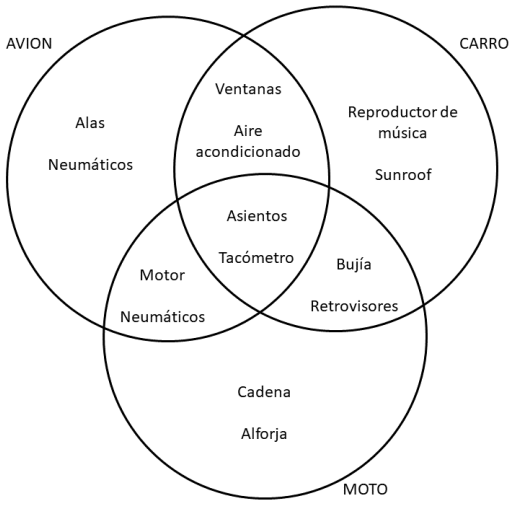

Figura 5. Diálogos.

El uso de estas funciones en un diálogo orientado a la tarea facilita la elaboración conjuntos y estructuras gramaticales de tales funciones, utilizando una gramática léxico funcional [19]. La elaboración de conjuntos y sus posibilidades gramaticales facilita la interpretación del diálogo y sirve para determinar la mejor manera de responder a una solicitud particular emanada de cualquier interlocutor [26], [27], [28].

Dentro de la construcción de un sistema de diálogo, es necesario contar con gestor que permita administrar los modelos de diálogo, utilizando su representación en un modelo estocástico. Dentro de las funciones de administración, se encuentran: terminar correctamente un diálogo iniciado, dar cabida a la creación de sub-diálogos y permitir la navegación entre los diálogos que cree un interlocutor. Sin importar qué sistema de diálogo se desarrolle, ya sea intencional o basado en pregunta-respuesta, es necesario contar con mecanismos que permitan medir su eficiencia.

SOBRE HADWARE

5.5. ARDUINO

Arduino es una plataforma de hardware de código abierto, basada en una sencilla placa de circuito impreso que contiene un controlador de marca “ATMEL” que cuenta con entradas y salidas analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación precessing. El dispositivo conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital controlando, sensores, alarmas, entre otros elementos [29].

Se seleccionó el Arduino debido a que es una herramienta muy compacta y completa ya que simplifica procesos de trabajar con micro controladores y ofrece algunas ventajas y características respecto a otros sistemas embebidos.

SOBRE EL SOFTWARE

5.6. HTK

HTK es un software dedicado para el diseño de sistemas de reconocimiento de voz. Este software permite utilizar diferentes metodologías de entrenamiento de los *HMMs* debido a que en el proceso de construcción de dos aplicaciones de reconocimiento automático de voz las cuales son: reconocimiento de palabras aisladas y reconocimiento de dígitos conectados [30].

HTK fue originalmente elaborado en la Universidad de Cambridge, donde se empezó a usar desde 1993 para el reconocimiento de habla de gran vocabulario. Entropic Research

Laboratory Inc. recibe todos los derechos de desarrollo de HTK en 1995 cuando fue creado como una sub-empresa de la universidad, entre los primeros desarrollos fueron la creación de los primeros modelos y librerías en código C [8], [30].

En la figura 6, se observa todas las herramientas del software, estas herramientas se llaman desde la consola de comandos de Windows.

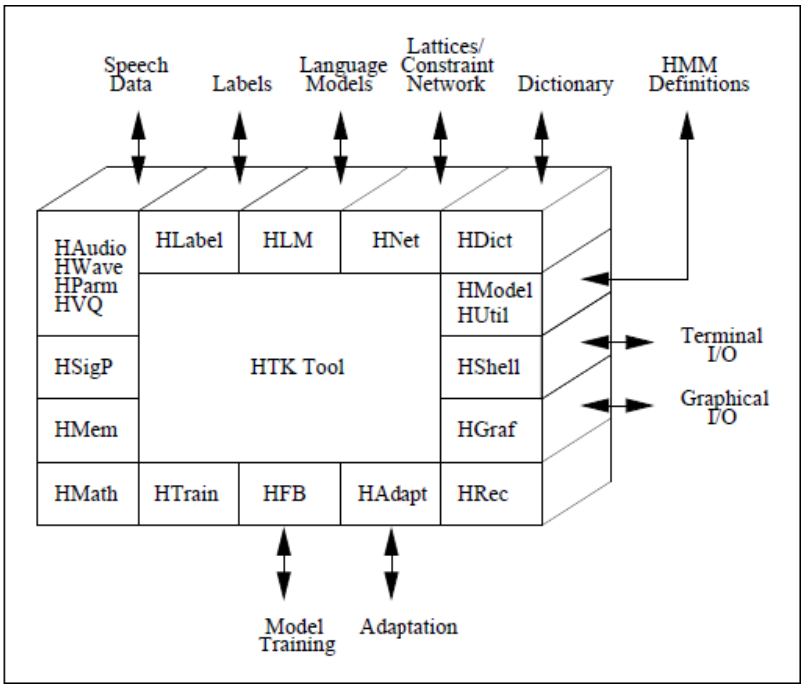

Figura 6. Arquitectura del software [30].

En la figura 7, se muestra un diagrama de bloques jerárquico, se puede observar 4 etapas que son fundamentales para la construcción de un modelo de entrenamiento, estas etapas son:

1. Preparación de los datos de entrenamiento,
2. Entrenamiento
3. Validación
4. Análisis.

Al final de todas estas etapas se tendrá un sistema el cual reconoce las palabras del modelo de entrenamiento.

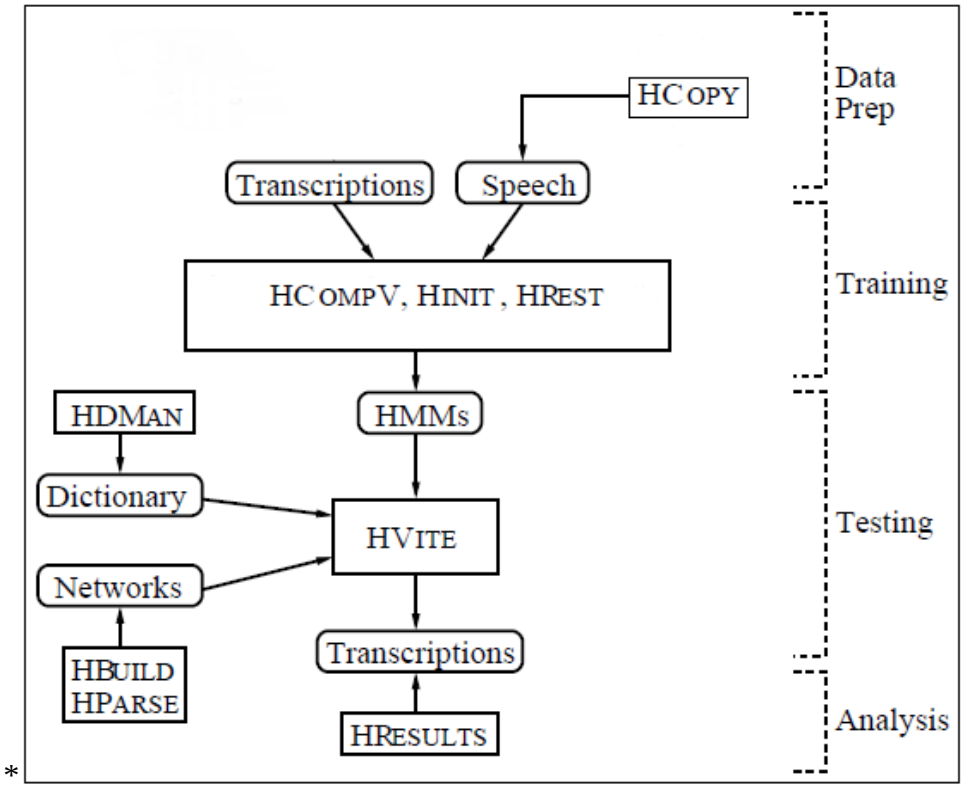

Figura 7. Estaciones de proceso HTK [8].

5.7. MATLAB

Matlab (*Matrix Laboratory*) es un lenguaje de alto nivel, diseñado para proveer facilidades de cálculos numéricos, visualización y programación en un entorno muy sencillo de utilizar. MATLAB ofrece un entorno de desarrollo integrado (IDE) muy versátil con un lenguaje de programación propio (lenguaje M). Proporciona una interfaz con líneas de comando para resolver problemas lineales y no lineales, además de otros experimentos en forma numérica. Utiliza una notación matemática simple para tratar los problemas y resolverlos [30].

Siendo MATLAB un intérprete de comandos, el cual permite efectuar cualquier operación con matrices de forma sencilla y rápida. Este sistema interactivo cuyo elemento básico es la matriz (y como caso particular, vectores o escalares), tanto de datos numéricos como de información no numérica. En el MATLAB todo elemento es considerado como una matriz: por ejemplo, un escalar es una matriz de (1x1); un vector es una matriz con solo una fila o una columna; una cadena de caracteres es una matriz fila de elementos (uno por letra), entre otros [30].

6. METODOLOGÍAS

En el capítulo anterior se indicó cuál es el objetivo principal de este trabajo, en este capítulo se presentará la metodología que se empleó para alcanzar dicho objetivo. En términos generales, este trabajo busca desarrollar un sistema de reconocimiento de voz y un sistema de diálogo basado en máquinas de estado finito para el control de una plataforma móvil.

Este trabajo condensa tecnología de diferentes campos como son el procesamiento de señales, el procesamiento de lenguaje natural, la electrónica digital y algunos principios de robótica aplicada.

En la figura 8 se presenta el diagrama del cómo se compone todo el sistema.

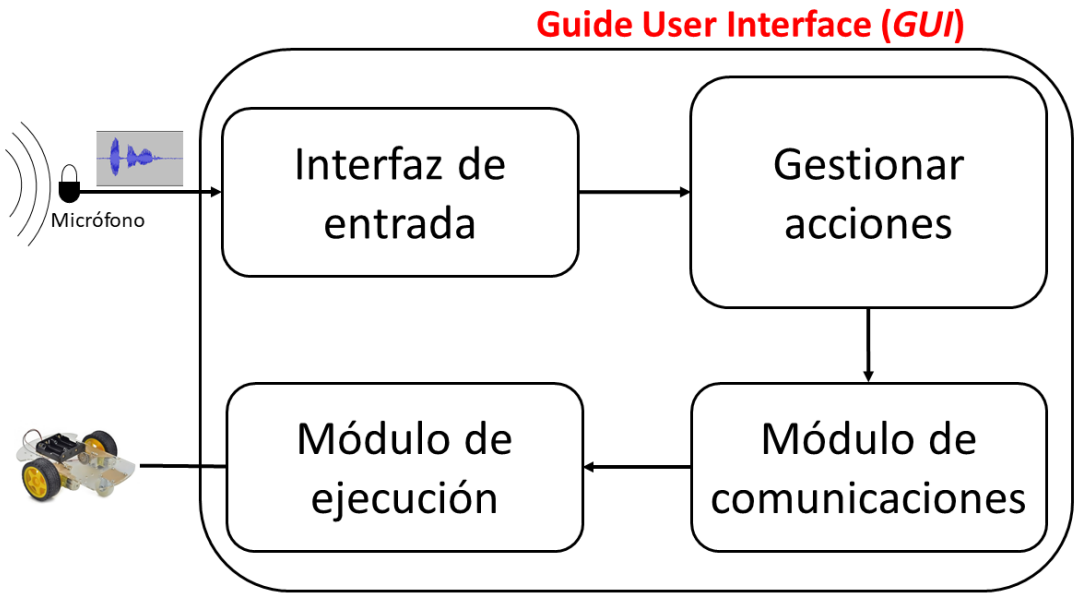

Figura 8. Diagrama del sistema.

A continuación, se presentará en detalle los segmentos presentados anteriormente con sus respectivas fases de la metodología que permitieron consolidar el empleo de distintas herramientas en el desarrollo de un único sistema.

6.1. Metodología para el desarrollo del sistema de reconocimiento automático de habla

La interfaz de entrada tiene como función realizar el reconocimiento de voz, es decir extraer información de las señales. En la figura 9 se presenta como se compone este segmento.

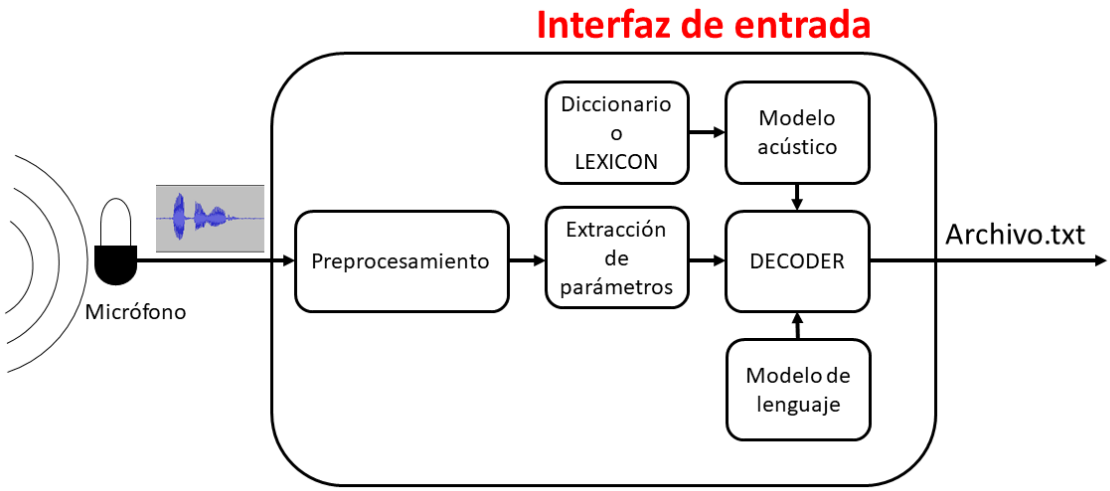

Figura 9. Arquitectura de la interfaz de entrada.

A continuación, se realiza una explicación más detallada de cada una de las fases mostradas en la figura 9.

6.1.1. Definición del *léxico*/diccionario.

La primera fase en el desarrollo del sistema de reconocimiento de voz es la determinación del *léxico* o *diccionario*. Se entiende como léxico al conjunto de todas las palabras que serán reconocidas por el sistema.

El primer paso en esta fase fue definir los comandos de voz con los que se controlará el sistema. Una vez estos han sido definidos, se procedió a identificar las palabras aisladas que componen estos comandos. El conjunto de todas estas palabras conformará el *léxico* del sistema de reconocimiento automático de voz.

En la Tabla 1 se presentan los comandos definidos para el control de la plataforma móvil.

No. De frase	Comando
1	Inicio
2	Pausa

3	Modo plataforma
4	Modo combate
5	Marcha uno
6	Marcha dos
7	Marcha tres
8	Reversa
9	Encender delanteras
10	Encender traseras
11	Encender luces
12	Apagar delanteras
13	Apagar traseras
14	Apagar luces
15	Girar 30 grados derecha
16	Girar 50 grados derecha
17	Girar 70 grados derecha
18	Girar 90 grados derecha
19	Girar 180 grados derecha
20	Girar 30 grados izquierda
21	Girar 50 grados izquierda
22	Girar 70 grados izquierda
23	Girar 90 grados izquierda
24	Girar 180 grados izquierda
25	Disparar

Tabla 1. Comandos para la plataforma móvil.

En la Tabla 2 se presenta el *léxico* que se definió para el sistema de reconocimiento de voz.

No. De palabras	Palabra
1	Inicio
2	Pausa
3	Girar
4	Treinta
5	Cincuenta
6	Setenta
7	Noventa

8	Ciento ochenta
9	Modo
10	Grados
11	Plataforma
12	Combate
13	Marcha
14	Uno
15	Dos
16	Tres
17	Derecha
18	Izquierda
19	Reversa
20	Encender
21	Apagar
22	Luces
23	Delanteras
24	Traseras
25	Disparar

Tabla 2. Léxico.

6.1.2. Generar una base de datos de comandos de voz para el entrenamiento y evaluación del sistema de reconocimiento de voz.

Una vez definido los comandos se procede a generar una base de datos de comandos de voz para el entrenamiento y evaluación del sistema de reconocimiento de voz. La base de datos son audios los cuales se categorizan en las clases como se observó en la Tabla 1, los cuales son grabaciones que se realizaron mediante el programa de **Audacity. S**e realizan 25 grabaciones por cada palabra, donde cada una de ellas cuentan con una frecuencia 16kHz, con extensión .wav como se presenta en la figura 10 una de las grabaciones, donde se pueden detallar algunas de las características que nos brinda el Audacity. Este proceso se realiza dos veces, debido a que la base de datos cuenta con dos locutores. Se añade un segundo locutor con el objetivo de que el sistema aumente su tasa de reconocimiento, y cuente con mejores estadísticas de reconocimiento con locutores desconocidos.

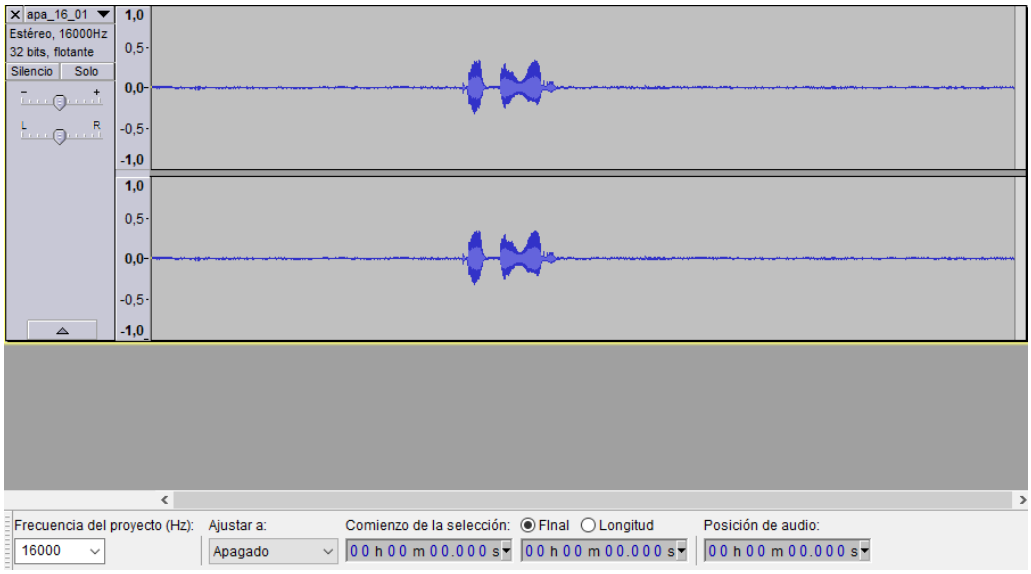

Figura 10. Señal de voz.

Para el entrenamiento de un sistema de reconocimiento de voz, se hace necesario identificar los intervalos sonoros y no como se presenta en la figura 11. Con base a estos intervalos se permitió que el entrenamiento de los modelos acústicos se haga de manera ajustada a las condiciones de las señales y de los comandos que se quieren identificar. Cabe aclarar que después de realizar varias pruebas se determinó que las grabaciones deben ser como mínimo de 4 a 5 segundos esto se debe a que si son muy cortas las grabaciones el software de HTK no procesa de manera correcta la información, además de que su parametrización no debe ser tan robusta y exacta ya que esto puede causar que no se reconozcan las palabras, ni sus tramas de silencio.

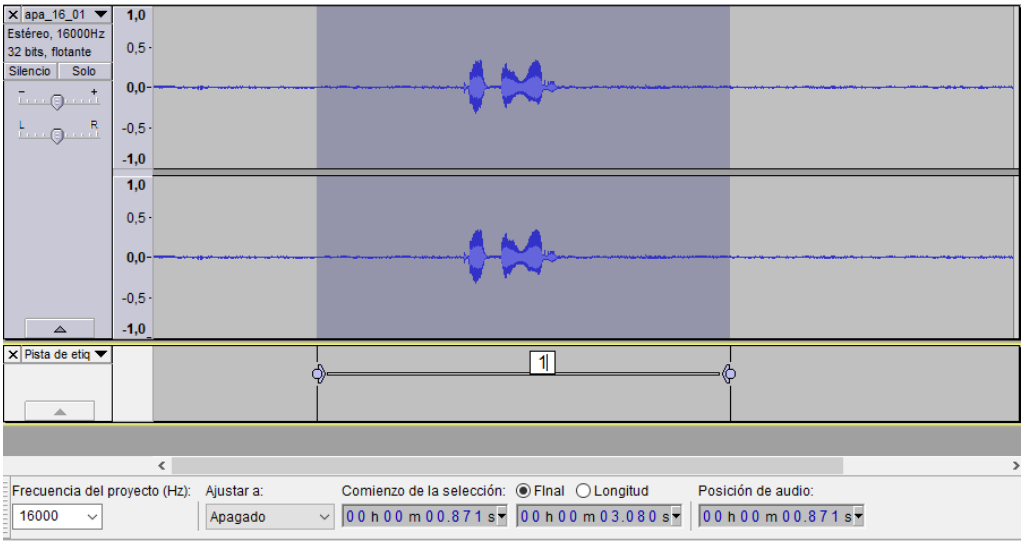

Figura 11. Señal de Audacity con intervalos sonoros y no sonoros.

Una vez que se presenta el resultado de la grabación de voz en el software de Audacity En la tabla 3 se presentan las palabras y sus características como lo son: etiqueta, frecuencia, si es mono o estéreo, entre otras cualidades del archivo.

Características de las grabaciones				
Formato	**Frecuencia**	**Canal**	**No. De bits**	**Tipo**
.wav	16kHz	Estéreo	32	Flotante

Tabla 3. Características de las grabaciones.

Una forma de obtener los valores temporales de los intervalos sonoros es empleando el software Audacity. **Ver anexo 1**. Este software ofrece la posibilidad de ver el tiempo de cada evento cuenta con un inconveniente y es que tiene al mostrar el tiempo seleccionado en segundos, ofrece pocas cifras decimales y esto hace perder precisión temporal al etiquetar las muestras. Por esta razón después de varias pruebas se determinó que ver la duración de los eventos en términos de muestras debido a que al analizarse en estas unidades nos permite tener mayor información sobre los eventos. Por ejemplo, el evento que aparece en la figura 12 ocurre en la muestra 13933.

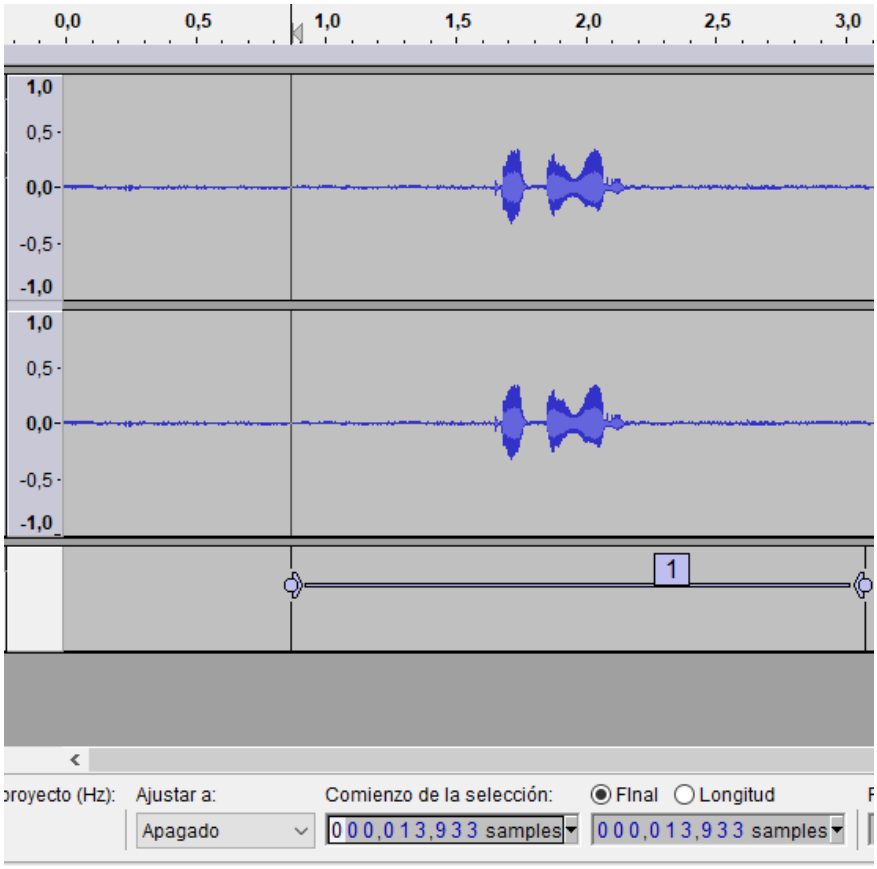

Figura 12. Ventana de tiempo en Audacity.

Una vez que se identifican los intervalos sonoros y no sonoros se procede a exportar las etiquetas por medio de Audacity, ya que este programa cuenta con una herramienta

llamada "*exportar etiquetas*", la cual permite extraer etiquetas de los intervalos a un fichero.txt. Cabe aclarar de que a pesar de que las señales se analizan en muestras estas etiquetas son exportadas en segundos. Audacity realiza la ecuación 3 para poder exportar las etiquetas en segundos, donde se exportan tan solo 3 datos, donde el primer dato es el inicio del intervalo sonoro, el segundo dato es el final de intervalo sonoro y el tercer dato denota el número de intervalos con la que cuenta la grabación.

$$t = \frac{13933}{16000} = 0{,}8703s$$

Ecuación 3. Ecuación de conversión de muestras a tiempo.

En la figura 13 se presenta el resultado de exportar las etiquetas.

```
0,870830        3,080000        1
```

Figura 13.Resultado de exportar etiquetas del software de Audacity.

6.1.3. Algoritmo reconocedor de intervalos sonoros y no sonoros.

Una vez que se obtuvieron todas las etiquetas de las grabaciones, se procedió a diseñar un algoritmo para la automatización de tareas.
Las tareas a automatizar fueron:

- **Conversión de muestras a centenas de nanosegundo**

  Una vez que se obtiene el valor de la muestra en segundos los momentos en los que ocurren los eventos sonoros este valor se multiplica por 10^7 para poder convertirlo a las unidades de centenas de nanosegundos de tal modo permitiendo conocer tanto al usuario como al software con más exactitud en momento inicio o finalizo el intervalo sonoro.

- **Ordenar los intervalos de tiempo y asignación de etiquetas.**

  Una vez que se realiza la conversión de unidades el paso a seguir fue la correcta estructuración de la etiqueta, y asignación de etiquetas, la cual consistió en darle la forma al archivo de siguiente manera, véase la figura 14. Donde se organiza y se etiqueta cada una de las tramas.

```
1  0,139    2,380000    1
2
```

```
1  0 13900000 sil
2  13900000 23800000 apa
3  23800000 45900000 sil
```

Figura 14. Forma ideal de los ficheros.

- **Cambio de extensión**

  El software a emplear recibe parámetros de entrada como lo son los archivos .lab, los cuales contienen las etiquetas en centenas de nano segundo y con una estructura ordenada como se especificó en el ítem anterior. Este algoritmo para finalizar condensa la conversión de unidades, y el la debida estructura en un fichero .lab. cómo se presenta en la figura 15, donde se puede diferenciar como cambia el archivo el cual se exporta de Audacity al ser ejecutado el algoritmo.

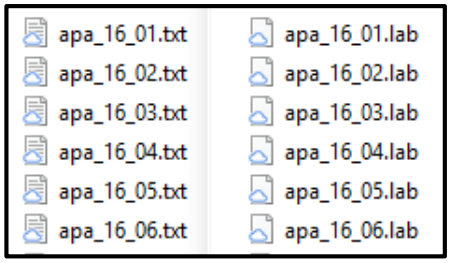

Figura 15. Resultados del algoritmo.

6.1.4. Análisis Acústico.

El análisis acústico se define como la conversión de una señal de onda a un vector de coeficientes cepstrales en la frecuencia de MEL (*MFCC*). Esto se realiza con el objetivo de la extracción de parámetros de las señales de voz.

La extracción de esta información se realizó bajo el comando ***HCopy*** propio del software *HTK* como se muestra en la figura 16. Este comando recibe la configuración de locación siendo este un script con extensión .txt el cual lleva por nombre **targetlist.txt** .Este script contiene las direcciones y formato de las señales a las cuales se va realizar el análisis acústico seguido a esta dirección se especifica el lugar en el cual se va a guardar el análisis acústico de cada señal junto al formato resultante como se presenta en la figura 17**.** También cuenta con el parámetro de entrada llamado "*analysis.conf",* fichero que determina los parámetros de análisis de las señales, como el ventaneo, el formato de las grabaciones, numero de coeficientes, para ver todos los parámetros véase la figura 18.

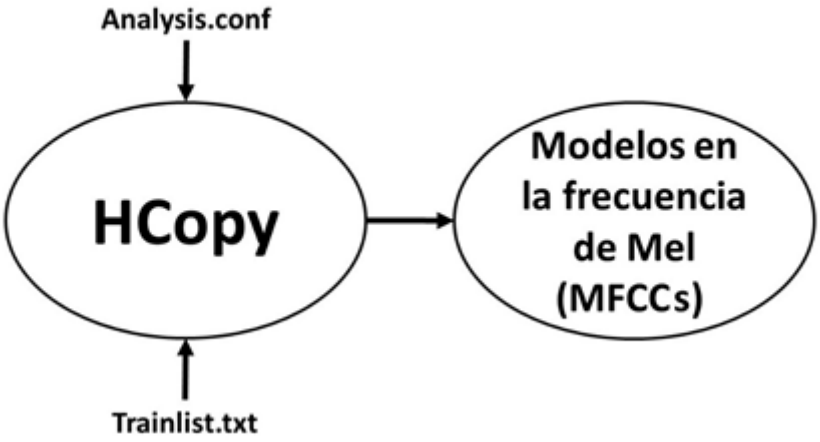

Figura 16.Arquitectura de la función del comando HCopy.

```
sig/apa_16_01_01.wav data/mfcc/apa_16_01_01.mfcc
sig/apa_16_01_02.wav data/mfcc/apa_16_01_02.mfcc
sig/apa_16_02_01.wav data/mfcc/apa_16_02_01.mfcc
sig/apa_16_02_02.wav data/mfcc/apa_16_02_02.mfcc
.
.
.
```

Figura 17. Arquitectura del targetlist.txt.

```
# Archivo de configuracion para analisis acustico
#
SOURCEFORMAT = WAV          # Define el tipo de formato del archivo de entrada
TARGETKIND = MFCC_0_D_A # Identifica los coeficientes que se van a usar
                            # el 0 identifica al coef c0
                            # la D identifca los coeficientes DELTA (es decir, las derivadas
                            # de [c0,c1,...,c12]
                            # la A identifica los coeficientes de ACELERACION (es decir,
                            # las segundas derivadas de [c0,c1,...,c12].
                            # la Z hace normalización cepstral CMN
                            # En total son 39 parámetros
# Unidad = 0.1 microsegundos (las mismas unidades en cientos de nanosegundos)
WINDOWSIZE = 250000.0       # 25 ms - ancho de una ventana de analisis
TARGETRATE = 100000.0       # 10 ms - periodicidad de la ventana (ni idea q es)

NUMCEPS = 12                # Numero de coefs cepstrales (van de c1 a c12)
USEHAMMING = T              # TRUE para el uso de ventana de Hamming
PREEMCOEF = 0.97            # Coeficiente para filtrado de pre-enfasis
NUMCHANS = 26               # Numero de canales de bancos de filtros
CEPLIFTER = 22              # Longitud del filtrado cepstral

# El fin
```

Figura 18. Fichero de configuración de parámetros para la extracción de parámetros.

6.1.5. Parametrización.

Esta fase tiene como objetivo inicializar la matriz de transiciones, donde para cada una de los modelos (es decir las palabras como silencio, modo, marcha, entre otras palabras) se generó un prototipo modelo *HMM* como el que se muestra en la figura 6, a la cual se le asignan valores nulos para su estado inicial.

En el prototipo se declara un vector de 39 posiciones que contienen los coeficientes *MFCC*, el nombre del prototipo *HMM* y el número de estados. Además, contiene unos valores de varianza iniciales iguales a 1 y de media iguales a cero para los estados.

Esta fase tiene como objetivo inicializar los parámetros con valores estimados de las palabras. Estos scripts deben estar contenidos en el directorio **model/proto/**. El script de tener el aspecto como el que se ilustra en la figura 19.

```
~o
<STREAMINFO> 1 39
<VECSIZE> 39<NULLD><MFCC_D_A_0><DIAGC> // DIMENSION DEL VECTOR Y EL TIPO DE COEFICIENTES A OBTENER
~h "si"                    // DESCRIPCION DE UN HMM llamado " sí" .
<BEGINHMM>
<NumStates> 6             //NUMERO DE ESTADOS
<State> 2                //ESTADO 2
<Mean> 39               //DA EL VECTOR MEDIO(EN UN ESPACIO DE OBSERVACION CON UNA DIMENSION DE 39)DE LA OBSERVACION ACTUAL
función
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39              //VARIANZA DEL ESTADO 2
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 3                 //ESTADO 3
<Mean> 39
0.0 0.0 (...) 0.0
<Variance> 39             //VARIANZA DEL ESTADO 3
1.0 1.0 (...) 1.0
<State> 4                //ESTADO 4
<Mean> 39
0.0 0.0 (...) 0.0
<Variance> 39           //VARIANZA DEL ESTADO 4
1.0 1.0 (...) 1.0
<State> 5              //ESTADO 5
<Mean> 39
0.0 0.0 (...) 0.0
<Variance> 39          //VARIANZA DEL ESTADO 5
1.0 1.0 (...) 1.0
<TransP> 6               //ES LA PROBABILIDAD DE TRANSICIÓN DEL ESTADO I AL ESTADO J
0.0 0.5 0.5 0.0 0.0 0.0  //LOS VALORES NULOS INDICAN QUE LA TRANSICIONES CORRESPONDIENTES NO ESTÁN PERMITIDOS.
0.0 0.4 0.3 0.3 0.0 0.0
0.0 0.0 0.4 0.3 0.3 0.0
0.0 0.0 0.0 0.4 0.3 0.3
0.0 0.0 0.0 0.0 0.5 0.5
0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figura 19. Plantilla modelo para la inicialización de modelos.

Una vez que se crearon todos los modelos para inicializar los modelos se ejecutó el comando ***HInit***. La arquitectura de este comando se presenta en la figura 20. Donde se presentan cuales son las entradas del comando y su respectiva salida

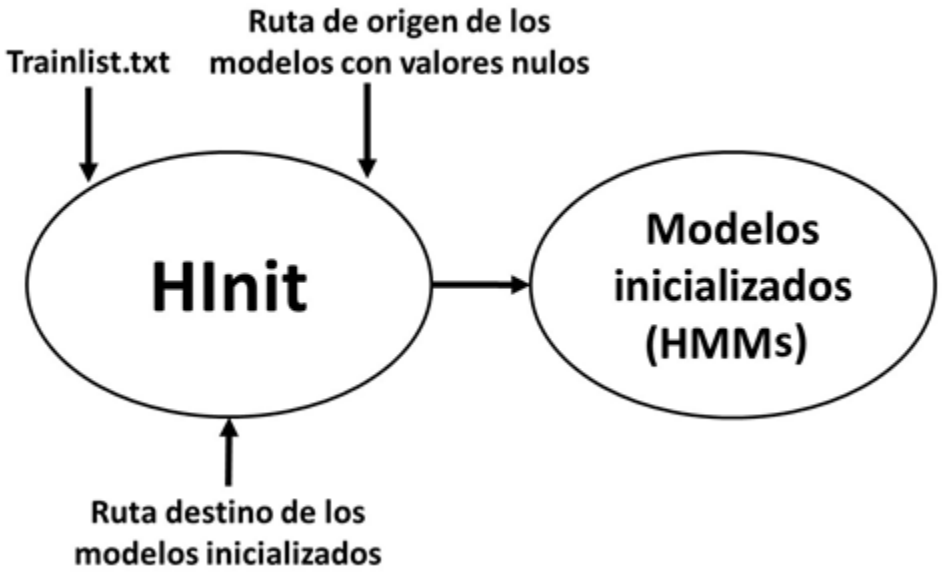

Figura 20. Arquitectura del comando HInit.

6.1.6. Parametrización global.

Una vez que se han inicializado los modelos del sistema fue necesario extraer una varianza global de todos los modelos. Esta función consistió en la normalización de los datos a partir del vector de varianza global multiplicado por un factor que se fija en el comando. En la figura 21 se presenta el comando que se utilizó para realizar la normalización de los datos.

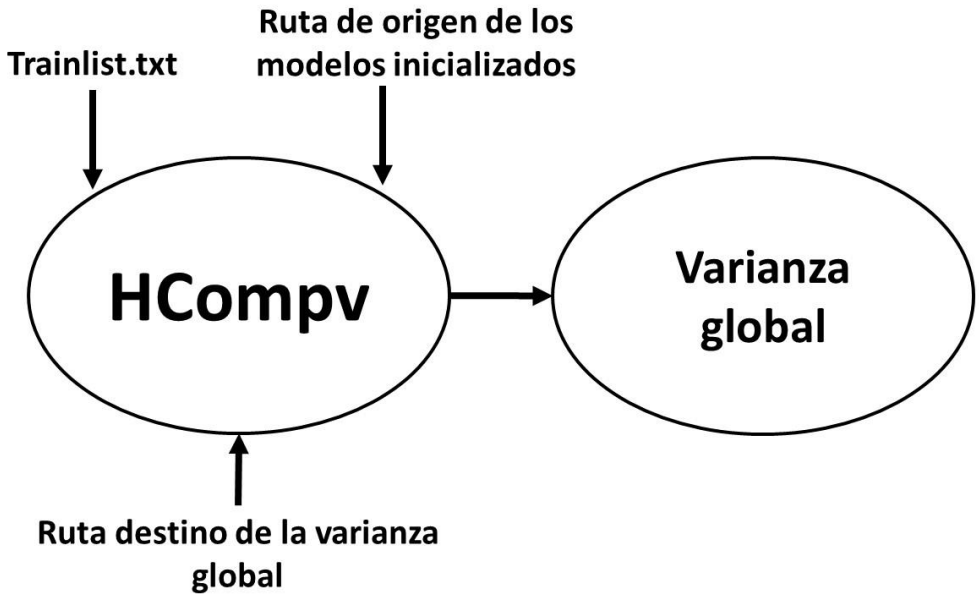

Figura 21. Arquitectura del comando HCompv. En la figuran se especifican también otros parámetros de entrada como son las rutas tanto de destino como de origen.

6.1.7. Reestimación de modelos.

Esta fase consiste en reestimar cada uno de los modelos ocultos de Markov. Esto se realiza para ajustar los valores de las matrices con ayuda de la varianza global donde el objetivo es mejorar la tasa de reconocimiento del sistema. El sistema que se diseñó se reestimó 7 veces, esto gracias a él comando ***HRest***, este comando se ilustra en la figura 22. Esta figura presenta las entradas del comando, donde se puede observar que además de la varianza global, está el trainlist.txt, que es el fichero que indica la dirección de los *MFCC*, la dirección del archivo el cual va tomar como base o modelo para la reestimación *"Ruta modelo i-1"* y por último la ruta destino del modelo en la cual se condensan los resultados del comando ejecutado "Ruta modelo i".

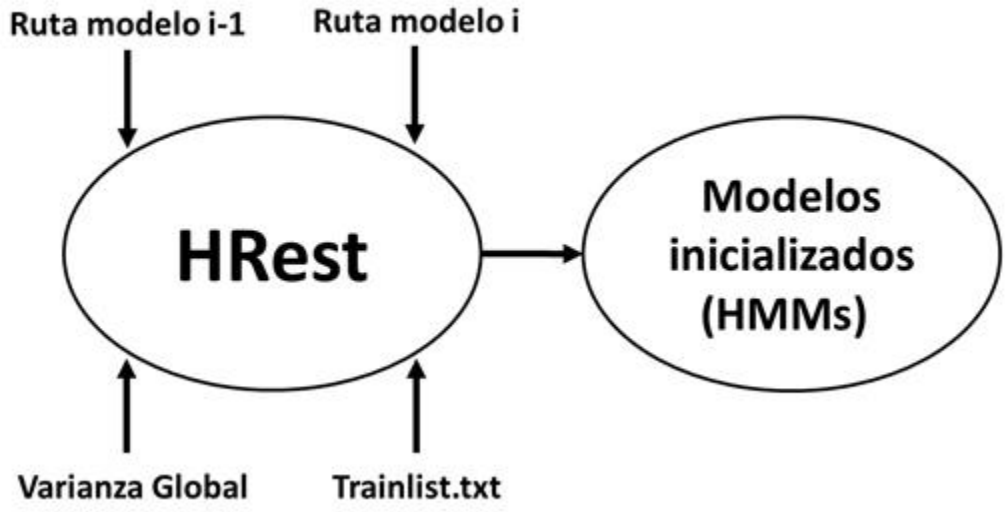

Figura 22. Arquitectura del comando HRest.

6.1.8. Modelos gramaticales

Una vez terminada la parte más técnica del proceso del reconocimiento del habla se procede a la parte gramatical, la cual consistió en definir en conjuntos las palabras, como se presenta en la figura 23.

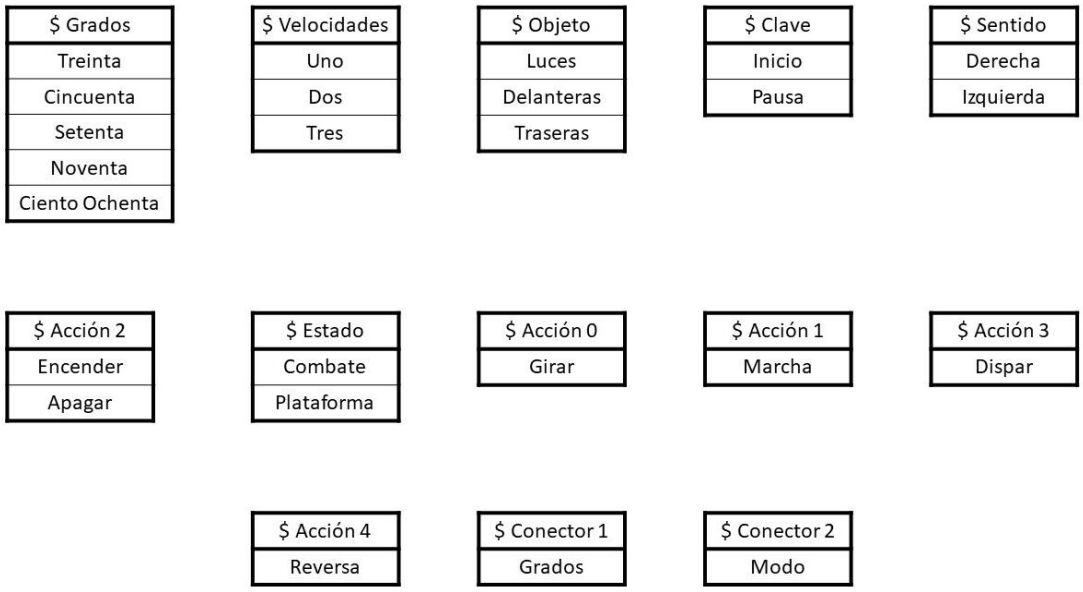

Figura 23. Conjuntos gramaticales.

Después de que se hayan definido los conjuntos gramaticales se procede a definir las estructuras gramaticales con estos conjuntos como se presenta en la figura 24, con el objetivo de que el sistema de reconocimiento de voz pueda concatenar las frases de manera correcta.

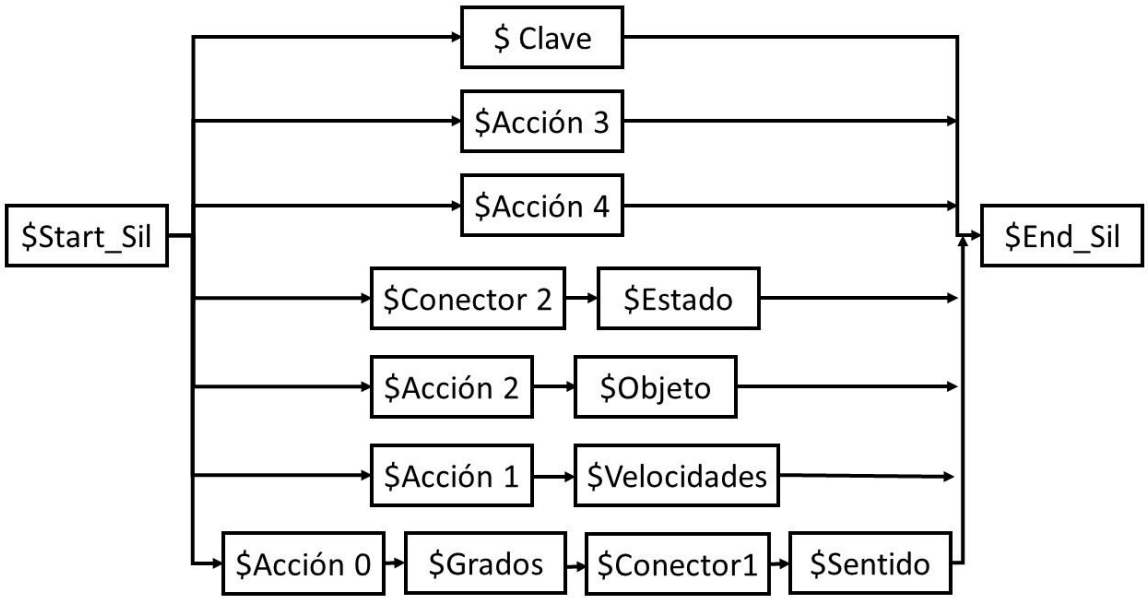

Figura 24. Estructuras gramaticales.

Finalmente, cuando se tienen los conjuntos y estructuras gramaticales se procede a ejecutar el comando ***HParse,*** comando cuya función principal es definir la red de trabajo, es decir guarda las posibilidades que hay entre los estados en un fichero llamado ***net.slf.*** Para observar con más detalle la arquitectura de este comando véase la figura 25.

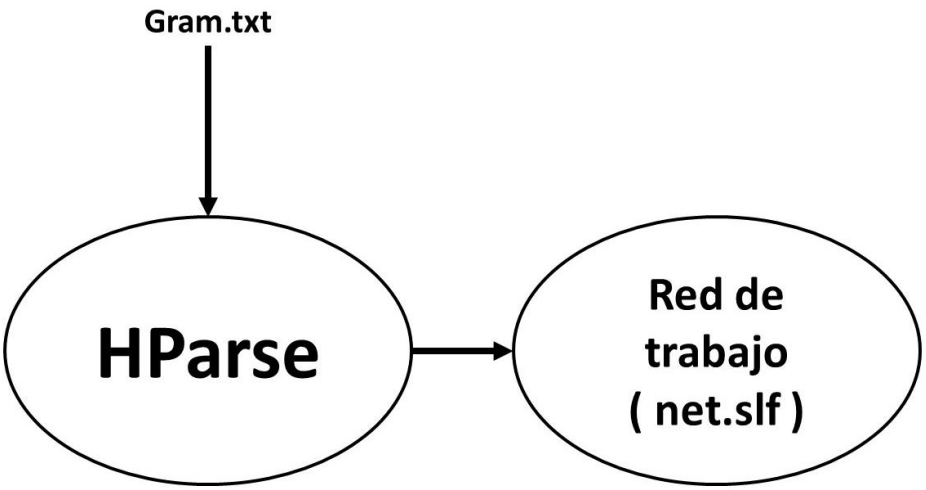

Figura 25. Arquitectura del comando HParse.

En ocasiones para probar el correcto funcionamiento de la red gramatical se procede a ejecutar el comando HSGen, para detallar más este paso opcional véase el anexo 1 pág. 13.

6.1.9. Reconocimiento de voz.

El reconocimiento de voz se puede realizar de tres diferentes maneras:

- Reconocimiento de un archivo
- Reconocimiento de varios archivos.
- Reconocimiento por directo (Micrófono).

La práctica de los diferentes tipos de reconocimiento se puede detallar en el anexo 1 en la página 13.

Para el sistema de reconocimiento de voz, se empleó el método de captura de señal de entrada directamente desde el micrófono y no desde archivos grabados en el computador ya que este nos permite de alguna manera evaluar el sistema en tiempo real. Esto se realiza con el objetivo de evaluar el sistema en diferentes condiciones (lugares no ideales).

6.2. Metodología para la implementación del gestor de diálogo

A continuación, en la imagen 26 se presenta la arquitectura del segmento gestionar acciones.

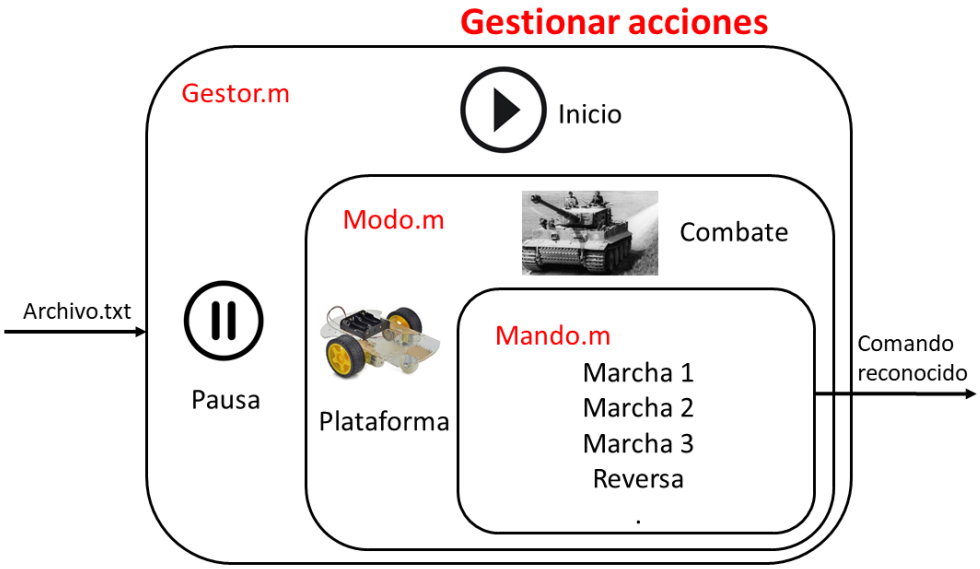

Figura 26. Arquitectura del gestor de diálogo.

A continuación, se presentan las fases realizadas para implementar el gestor de acciones.

6.2.1. Diseño de la máquina de estados finitos

El primer paso para la implementación del gestor de diálogo fue diseñar y definir la máquina de estados finitos, para poder determinar las posibles transiciones de cada uno de los estados, permitiendo de este modo evitar ambigüedades y evidenciar condiciones del sistema.
En la figura 27 se presenta el diseño de la máquina de estados para el sistema completo. Cabe aclarar que todos los estados que se encuentran dentro de la circunferencia cuentan con cuatro posibles transiciones:

- Que se repita el comando y por lo tanto se queda en el mismo estado.
- Que pase a otro estado del mismo tipo de funcionamiento, es decir otra acción que pertenece al modo plataforma.
- Pausar el sistema.
- Cambiar de modo (Mod.Plataforma||Mod.Combate).

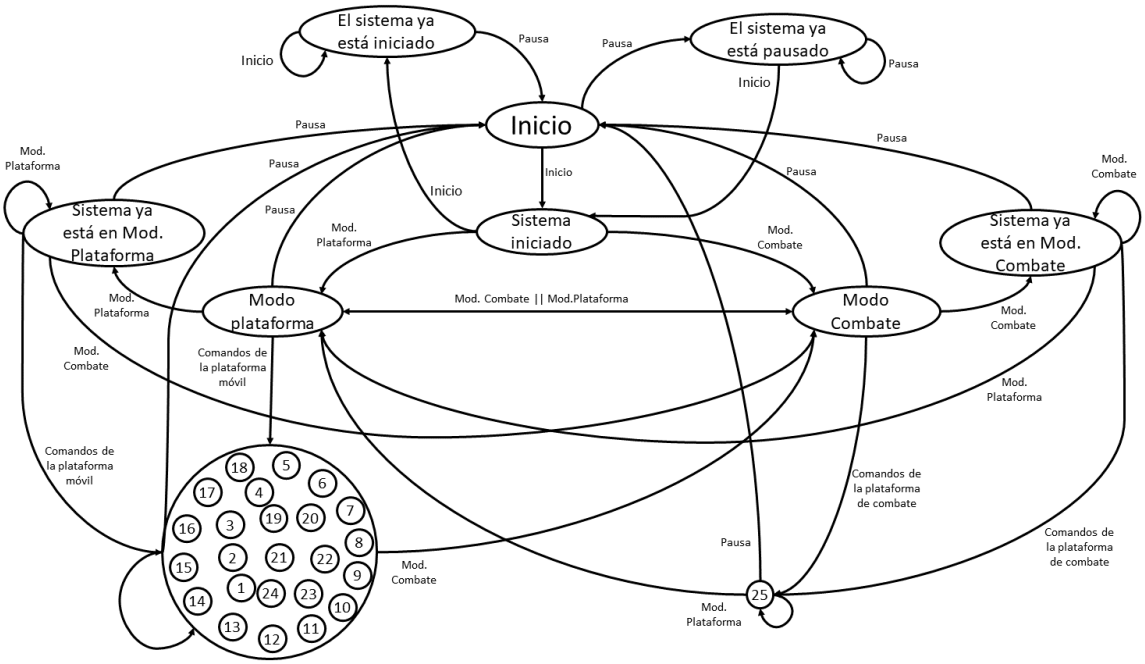

Figura 27. Máquina de estados finito del sistema.

6.2.2. Diseño de una interfaz gráfica de usuario (GUI)

Se procede a diseñar un entorno gráfico sencillo y amigable con el usuario, esto con el fin de que pueda de manera fácil manipular la interfaz. En la figura 28 se presenta la interfaz gráfica de usuario donde cuenta con apoyos como lo es un breve manual y la lista de los comandos, con el objetivo de que el usuario tenga claro cuáles son los comandos que puede ejecutar de manera correcta.

Figura 28. Interfaz gráfico de usuario (GUI).

En la figura 29 se presenta el manual, el cual hace una breve introducción del funcionamiento del GUI.

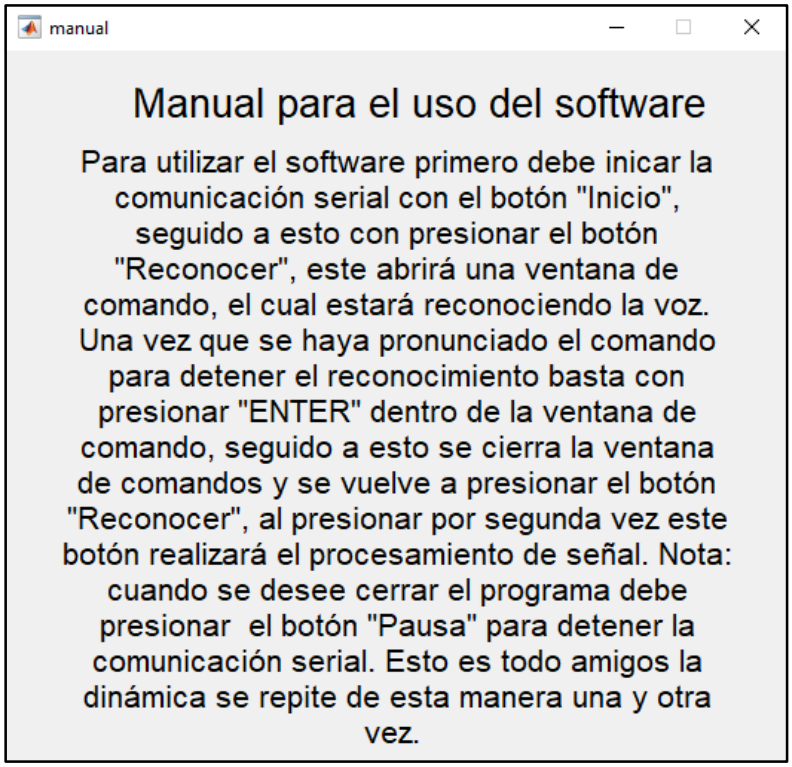

Figura 29. Manual para la interfaz gráfica de usuario.

En la figura 30 se presenta el flujograma de los comandos, esto se realiza con el objetivo de garantizar que el usuario interactúe de manera correcta con el gestor de diálogo.

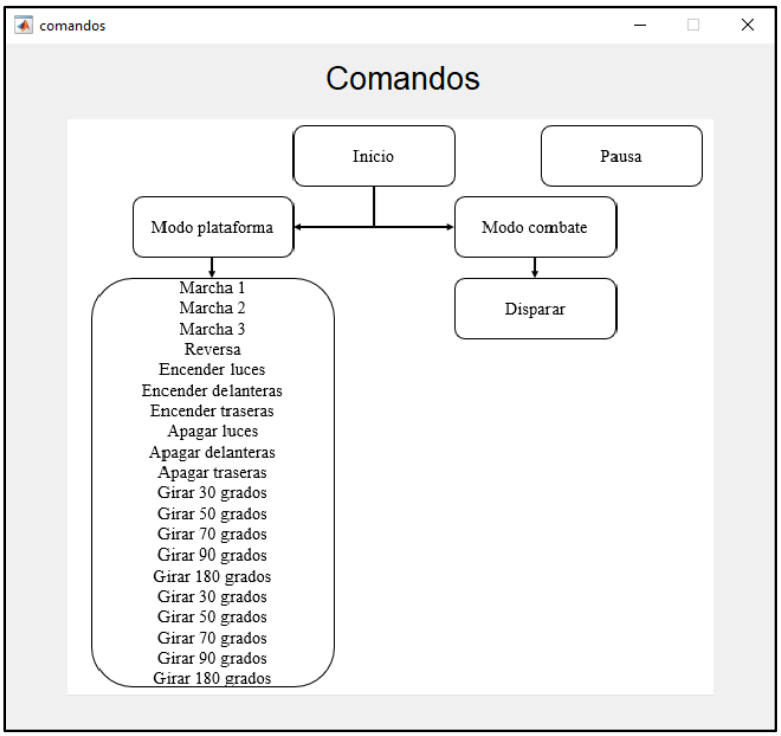

Figura 30. Flujograma del gestor de diálogo.

6.2.3. Diseño e implementación de funciones con base a la máquina de estados para el GUI

Una vez que se creó un diseño y vez establecida la máquina de estados del gestor de diálogo, se procedió a diseñarla mediante funciones. El diseño de las funciones se realizó bajo el recurso de variables auxiliares y comparaciones que se enfocan en cumplir la dinámica del sistema.
El sistema funcionó de manera efectiva bajo tres funciones:

- ***Gestor.m***

Esta función fue la encargada de determinar si el sistema iniciaba, se pausaba o por el contrario ya se había iniciado o se había pausado.
En la figura 31 se presenta la arquitectura de la función, ilustrando cuáles son los parámetros de entrada y sus respectivas salidas.

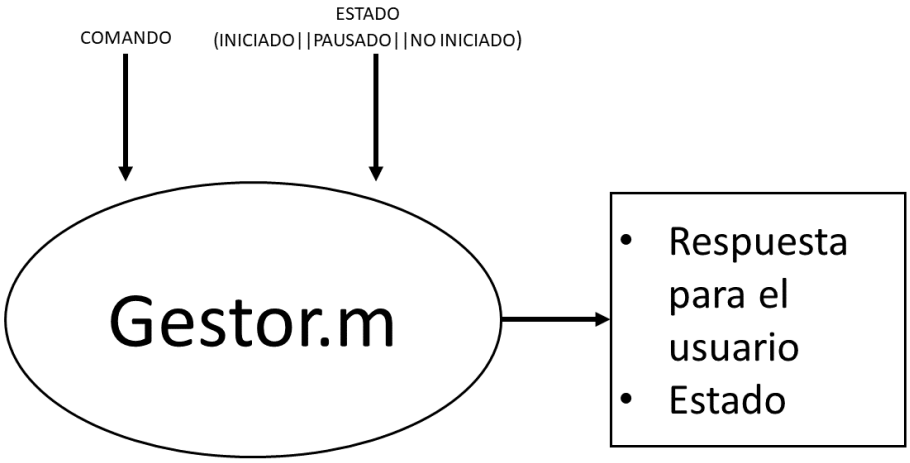

Figura 31. Arquitectura del gestor.m.

Para entrar en más detalles se presenta la máquina de estados correspondiente a la función. Véase la figura 32.

Figura 32. Máquina de estados finitos de la función gestor.m.

- ***Modo.m***

  Esta función se encargó de verificar que una vez que el sistema se fuera iniciado, este permitiera definir bajo qué modo de plataforma iba a trabajar (Modo plataforma||Modo combate).

En la figura 32 se presenta la arquitectura de la función, ilustrando cuáles son los parámetros de entrada y sus respectivas salidas.

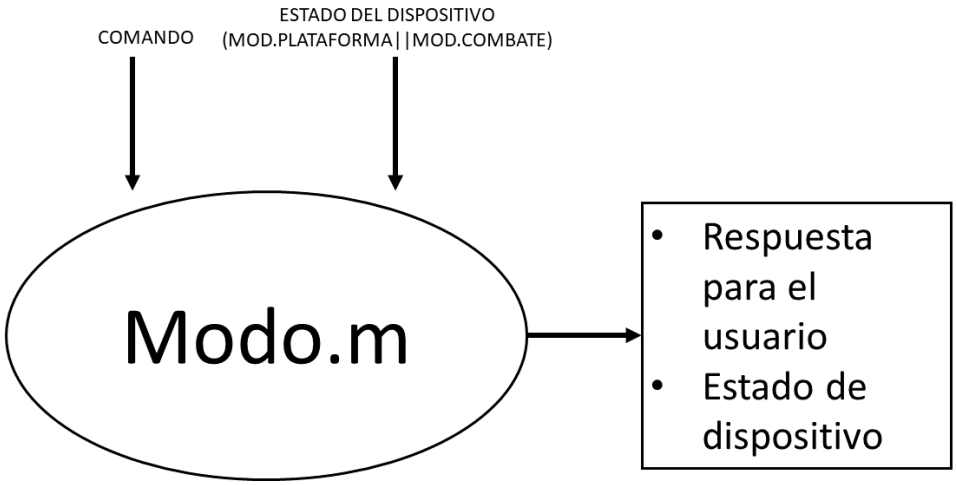

Figura 33. Arquitectura de modo.m.

Para entrar en más detalles se presenta la máquina de estados correspondiente a la función. Véase la figura 34.

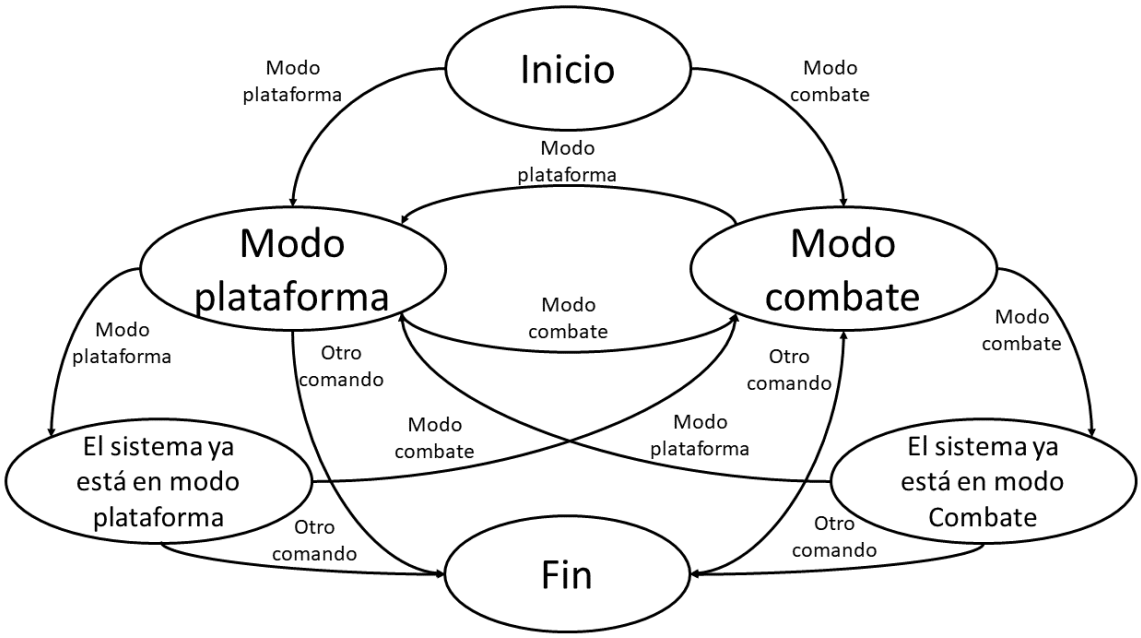

Figura 34. Máquina de estados finitos de la función gestor.m.

- ***Mando.m***

  Por último, esta función se encargaba de revisar que el sistema se hubiera iniciado, y que estuviera en el modo correspondiente al comando si cumplía estas condiciones ejecuta la acción de lo contrario emite un mensaje justificando la razón por la que no ejecutó la acción.
  En la figura 35 se presenta la arquitectura de la función, ilustrando cuáles son los parámetros de entrada y sus respectivas salidas.

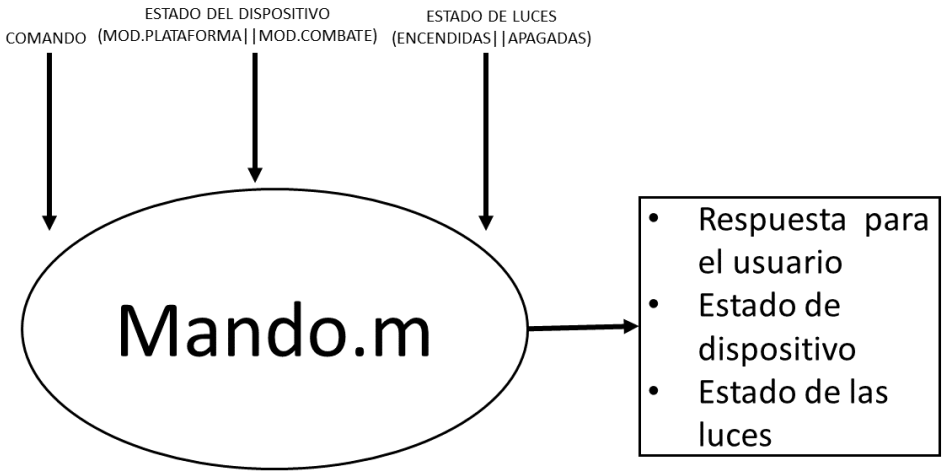

Figura 35. Arquitectura de mando.m.

Para entrar en más detalles se presenta la máquina de estados correspondiente a la función de ***mando.m*** en la figura 36, donde se presenta la máquina de estados que obedece a la función ***mando.m***, donde la transición de cada uno de los estados depende del modo en que se encuentre y si y solo si pertenecen a grupo de comandos del modo en que se encuentre como se presentó en la figura 30.

NOTA: Cabe aclarar que como se explicó anteriormente en la máquina de estados del sistema completo todos los estados que se encuentran dentro de la circunferencia tienen cuatro posibles transiciones.

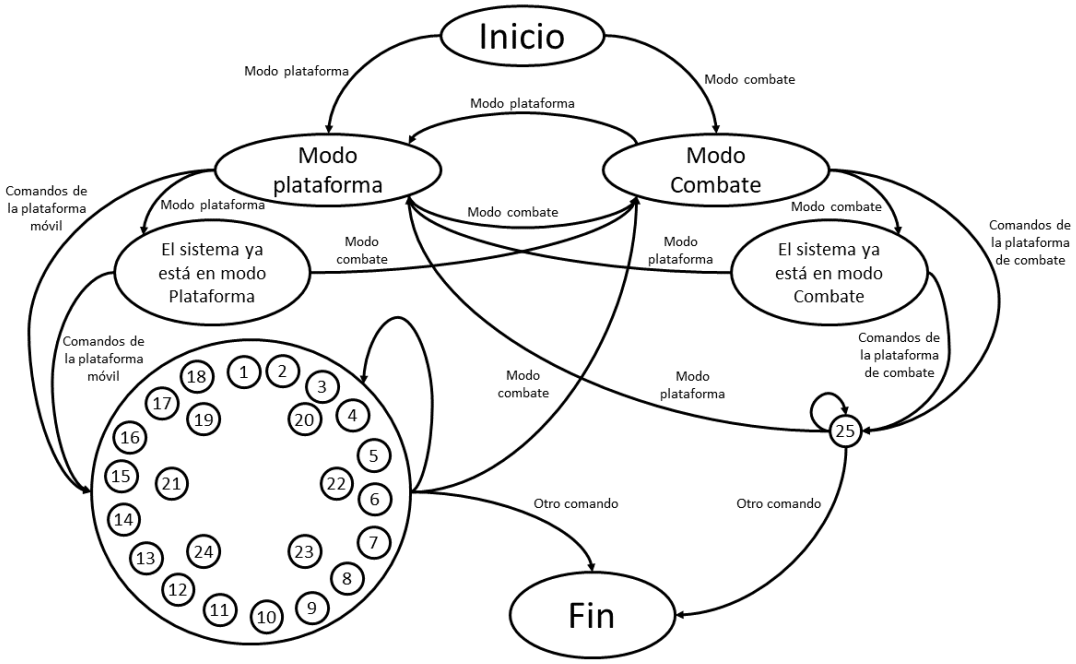

Figura 36. Máquina de estados finitos de la función gestor.m.

Una manera de visualizar el modo correcto de ejecutar los comandos es ver su estructura jerárquica como se presenta en la figura 37.

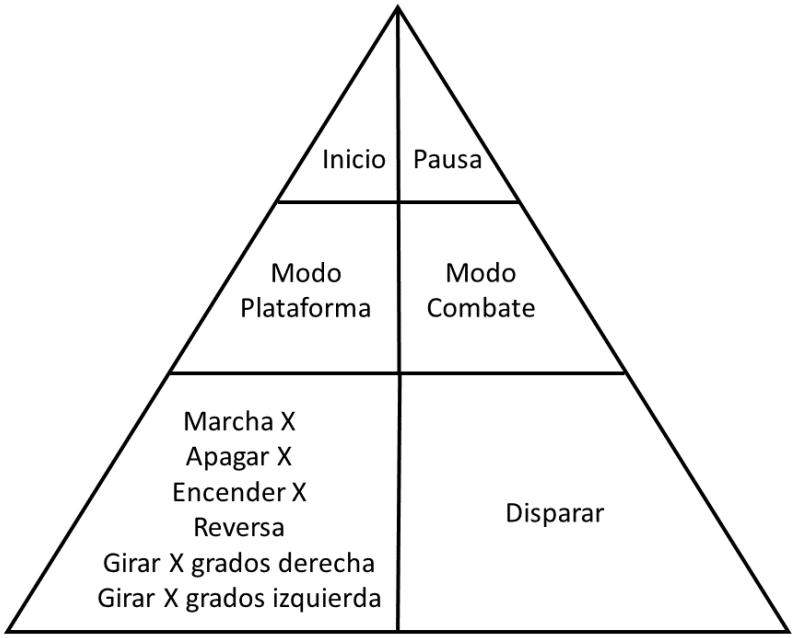

Figura 37.Jerarquía de comandos.

6.3. Metodología para la plataforma móvil

6.3.1. Establecer comunicación serial

Esta fase del proyecto consistió en garantizar la comunicación desde el *GUIDE* de Matlab al dispositivo de control, mediante comunicación serial. En la figura 38 se presenta el esquema de comunicación del ordenador con el sistema de control, es decir, el Arduino y a su vez especifica el medio de comunicación (Comunicación serial o más conocida como RS232).

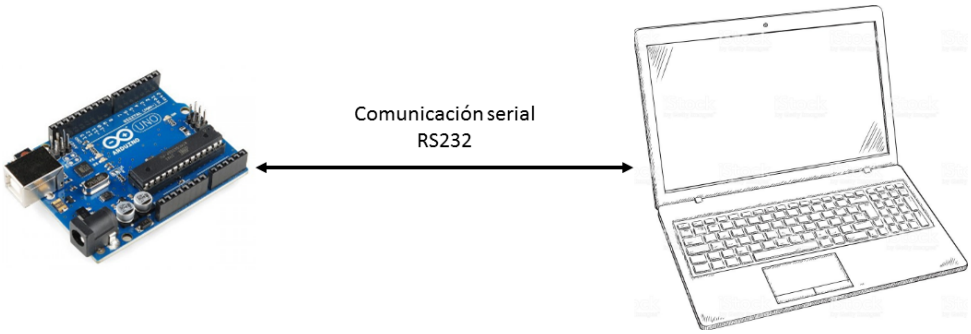

Figura 38. Esquema de comunicación.

6.3.2. Implementación de máquinas de estados en el Arduino

Una vez que se ha garantizado la comunicación entre el *GUIDE* y el sistema de control se procede a realizar la máquina de estados para el control de periféricos. Para esto fue necesario enviar un carácter cuando el comando es un acción sobre periféricos o hardware desde el *GUI*, para ser más específico desde la función ***mando.m*** . Estos caracteres se asignaron bajo un criterio libre.

Comandos	Carácter asignado
girar 30 grados izquierda	v
girar 50 grados izquierda	b
girar 70 grados izquierda	n

girar 90 grados izquierda	m
girar 180 grados izquierda	1
girar 30 grados derecha	k
girar 50 grados derecha	l
girar 70 grados derecha	z
girar 90 grados derecha	x
girar 180 grados derecha	c
encender luces	i
apagar luces	d
encender delanteras	a
encender traseras	p
apagar delanteras	h
apagar traseras	f
disparar	q
reversa	e
marcha 1	r
marcha 2	t
marcha 3	y

Tabla 4. Asignación de caracteres a comandos que representan acción en periféricos.

Luego de haber asignado un caracter a cada uno de los comandos se implementa un condicional ***switch*** en el Arduino en el cual se especifican las acciones a ejecutar dependiendo del carácter que reciba.

7. RESULTADOS

7.1. Resultados sobre el reconocimiento de voz.

Para conocer cuál fue la efectividad y los resultados del sistema se realizó una prueba rigurosa donde se evalúo el sistema con cada uno de los locutores. Esta evaluación consistió en que cada uno de los locutores pronunciaba 10 veces cada uno de los comandos de la plataforma móvil. En la tabla 4 se presentan el recuento de número de veces pronunciadas, el recuento del número de fallas entre otras características del proceso de evaluación y en la Tabla 5 se consolidaron los resultados y la tasa de efectividad del primer locutor. Por ejemplo, en la frase número 4 "*girar 50 grados izquierda*", se cuenta con 4 palabras en el comando (No. de palabras de la frase), dónde se pronunció 10 veces el comando (No. de veces que se pronunció), una vez que se pronunció 10 veces el comando se tiene un recuento de cuántas palabras se pronunciaron, en este caso 40 palabras (Recuento de palabras - 10 x 4 = 40 palabras), en el cual se tuvo un error del 10%, es decir, tuvo error al reconocer 4 palabras del comando.

No. de frase	Comandos	No. de palabras del comando	No. de veces que se pronunció	Recuento de palabras	% de error
1	inicio	1	10	10	0%
2	pausa	1	10	10	0%
3	girar 30 grados izquierda	4	10	40	0%
4	girar 50 grados izquierda	4	10	40	10%
5	girar 70 grados izquierda	4	10	40	0%
6	girar 90 grados izquierda	4	10	40	0%
7	girar 180 grados izquierda	4	10	40	10%
8	girar 30 grados derecha	4	10	40	0%

9	girar 50 grados derecha	4	10	40	0%
10	girar 70 grados derecha	4	10	40	10%
11	girar 90 grados derecha	4	10	40	0%
12	girar 180 grados derecha	4	10	40	10%
13	modo combate	2	10	20	10%
14	modo plataforma	2	10	20	0%
15	encender luces	2	10	20	2,5%
16	apagar luces	2	10	20	0%
17	encender delanteras	2	10	20	0%
18	encender traseras	2	10	20	0%
19	apagar delanteras	2	10	20	0%
20	apagar traseras	2	10	20	0%
21	disparar	1	10	10	10%
22	reversa	1	10	10	5%
23	marcha 1	2	10	20	0%
24	marcha 2	2	10	20	0%
25	marcha 3	2	10	20	0%
TOTAL			250	660	4,09%

Tabla 5. Evaluación del sistema para el locutor 1.

# de palabras pronunciadas (10 por c/u)	# de palabras reconocidos	Tasa de reconocimiento
660	633	95.9090%

Tabla 6. Resultados del sistema de reconocimiento de voz para el locutor 1.

En la tabla 6 se presentan el recuento de número de veces pronunciadas, el recuento del número de fallas entre otras características del proceso de evaluación para el segundo locutor y en la Tabla 7 se consolidaron los resultados y la tasa de efectividad del segundo locutor.

No. de frase	Comandos	No. de palabras del comando	No. de veces que se pronunció	Recuento de palabras	% de error
1	Inicio	1	10	10	0%
2	Pausa	1	10	10	0%
3	girar 30 grados izquierda	4	10	40	0%
4	girar 50 grados izquierda	4	10	40	0%
5	girar 70 grados izquierda	4	10	40	10%
6	girar 90 grados izquierda	4	10	40	0%
7	girar 180 grados izquierda	4	10	40	20%
8	girar 30 grados derecha	4	10	40	0%
9	girar 50 grados derecha	4	10	40	10%
10	girar 70 grados derecha	4	10	40	0%
11	girar 90 grados derecha	4	10	40	0%
12	girar 180 grados derecha	4	10	40	20%

13	modo combate	2	10	20	10%
14	modo plataforma	2	10	20	0%
15	encender luces	2	10	20	0%
16	apagar luces	2	10	20	5%
17	encender delanteras	2	10	20	5%
18	encender traseras	2	10	20	5%
19	apagar delanteras	2	10	20	5%
20	apagar traseras	2	10	20	5%
21	Disparar	1	10	10	33,33%
22	Reversa	1	10	10	20%
23	marcha 1	2	10	20	10%
24	marcha 2	2	10	20	10%
25	marcha 3	2	10	20	10%
TOTAL			250	660	7,4242%

Tabla 7. Evaluación del sistema para el locutor 2.

No. de palabras pronunciadas (10 por c/u)	No. de palabras reconocidos	Tasa de reconocimiento
660	611	92,5757%

Tabla 8. Resultados del sistema de reconocimiento de voz para el locutor 2.

Una vez que se realizó la evaluación del sistema de reconocimiento de voz de manera independiente, se dio paso a calcular la tasa de reconocimiento total, la cual consistió en promediar los resultados de los locutores. Estos resultados se consolidaron en la tabla 8. Donde cabe destacar que la tasa de reconocimiento del sistema con dos

locutores es del 94 %. El error del 6% puede deberse a que la evaluación del sistema se hizo en un ambiente controlado diferente al establecido.

Tasa de reconocimiento	
Locutor 1	95.9090%
Locutor 2	92.5757%
Total	94,2423%

Tabla 9. Tasa de reconocimiento total del sistema.

A continuación, se presenta el enlace del video donde se puede observar el funcionamiento del sistema de reconocimiento de voz.

- Reconocimiento por ventana de comandos mediante HTK: https://youtu.be/dgWsS_MoQXE

7.2. Resultados del gestor de diálogo

A continuación, se presenta el enlace del video donde se puede observar el funcionamiento del gestor de diálogo.

- Gestor de diálogo para el sistema reconocedor de voz:
  https://youtu.be/5FmKWZrQaJI
  Donde se presenta como se realizó un seguimiento del flujograma que se presentó en la figura 30 con el objetivo de demostrar la efectividad de la GUI.

7.3. Resultados del proyecto

A continuación, se presenta el enlace a un video donde se puede observar el funcionamiento del sistema completo.

- Control de plataforma móvil mediante reconocimiento de voz e interfaces de habla: https://www.youtube.com/watch?v=g3SnzrLar-4

8. CONCLUSIONES Y TRABAJOS A FUTURO

- Se logra desarrollar un sistema de reconocimiento de voz de habla aislada para el reconocimiento de comandos por voz para el control de una plataforma móvil.

- Se logró la implementación del gestor de diálogo para la interacción entre el usuario y la plataforma móvil.

- Se logró establecer la comunicación entre el gestor de diálogo y la plataforma móvil.

- Se desarrolló un sistema de reconocimiento de voz y un sistema de diálogo basado en máquinas de estado finito para el control de una plataforma móvil.

- Se evidenció que al tener una base de datos con dos locutores el sistema logra acaparar más parámetros, permitiendo de este modo aumentar la tasa de reconocimiento con respecto a locutores ajenos a la base de datos.

- Una vez diseñado y desarrollado un sistema de reconocimiento de voz, estos sistemas suelen tener la particularidad de que si este se ejecuta en otro ordenador tiene una posibilidad del 90% de que sea erróneo el resultado, debido a que cada micrófono cuenta con una ganancia diferente (En su mayoría) además de que cabe aclarar que los ordenadores no suelen tener las mismas propiedades restringiendo aún más el sistema.

- La automatización de procesos tan mecánicos como lo es el etiquetado, grabación de voz, entre otros, es de vital importancia, ya que en procesos mecánicos nunca deja de existir el error por factor humano, donde siempre varía el % error.

- La varianza global del sistema es uno de los factores más importantes de todo el proyecto ya que es con este fichero con el cual se trazan los modelos y se fijan los parámetros de variación de cada uno de los modelos.

- Emplear FeedBack o sistemas de retroalimentación permiten que el sistema de una respuesta clara sobre los errores que comete el usuario cuando emplea el software.

- La implementación de gestores de diálogo en sistemas de control permite garantizar de una manera didáctica la correcta manipulación del software o hardware.

- El gestor de diálogo es una interfaz gráfica de usuario que permite informar o direccionar al usuario; los gestores de dialogo se implementan mediante máquinas de estados, garantizando de este modo ambigüedades en el sistema.

- No fue posible comunicar el sistema de reconocimiento de voz (software) con el Arduino debido a que el dispositivo bluetooth no era compatible con el equipo, a pesar de que se actualizaron los drivers y se cambió la tarjeta bluetooth. Por lo que se propuso trabajar mediante comunicación serial, con el objetivo de poder llegar al mismo objetivo.

Proyectos que podrían implementarse en un futuro

- Un sistema de reconocimiento de voz que emplee un gestor de diálogo para traducir las palabras para personas con discapacidad auditiva.
- Plataformas interactivas para la enseñanza con diferentes temáticas.
- Control de drones mediante reconocimiento de voz.
- Control domótico aplicado con sistemas embebidos.
- Control de tareas mediante un sistema de reconocimiento de voz.
- Control de enjambres robóticos mediante un sistema reconocedor de voz.
- Control de una matriz de leds mediante un sistema reconocedor de voz.

9. BIBLIOGRAFÍA Y/O REFERENCIAS

PRIMER DOCUMENTO

[1] "Importancia de la voz", [Online], disponible en: https://www.importancia.org/voz.php , [Fecha de consulta: 15 Febrero 2017] .

[2] "Introducción a las tecnologías del habla", [Online], disponible en: http://physionet.cps.unizar.es/~eduardo/investigacion/voz/intro.html , [Fecha de consulta: 15 Febrero 2017]

[3] Rua S. L. C. Sistema de reconocimiento de voz aislada utilizando *Arduino Micro* y *Bitvoicer Server* para control domótico simulado, Universidad Tecnológica de Pereira, [Fecha de consulta:10 Enero del 2017] Disponible en: http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/7102/6213822R894.pdf?sequence=1&isAllowed=y

[4] S Young, et al. "HTKBOOK", Microsoft Corporation Cambridge University Engineering Department. (Ed 3.4), 2006 [Fecha de consulta: 30 Julio 2016] Disponible en: http://speech.ee.ntu.edu.tw/homework/DSP_HW2-1/htkbook.pdf

[5] Ivan O V. 2005, Aplicaciones en Reconocimiento de Voz Utilizando HTK, Universidad Javeriana Bogotá [Fecha de consulta: 9 Enero 2017] http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis95.pdf

[6] Martínez A. J. L. diseño e implementación de un sistema de reconocimiento de voz mediante *RASPBERRY PI,* Universidad Tecnológica de Pereira[Fecha de consulta:9 Enero del 2017] Disponible en: http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/7432/6213822M385d.pdf?sequence=1&isAllowed=y

[7] Sistema VLSC ,sistema inteligente de reconocimiento de voz para la traducción del lenguaje verbal a la lengua de señas colombiana, Universidad Pedagógica Nacional[Fecha de consulta: 15 Enero 2017] Disponible en: http://www.ribiecol.org/embebidas/congreso/2008/ponencias/35.pdf

[8] Alzate V. E. A. Desarrollo de un sistema de reconocimiento de comandos de habla aislada para el control automático de un vehículo simulado, Universidad Tecnológica de Pereira[Fecha de consulta:14 Enero del 2017] Disponible en: http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/7431/6213822A478.pdf?sequence=1

[9] Yueng-Tien Lo Tien Lo, Hidden Markov Toolkit (HTK) Installation, Department of Computer Science & Information Engineering National Taiwan Normal University [Fecha de consulta:15 Enero 2017] Disponible en:

http://berlin.csie.ntnu.edu.tw/Courses/Speech%20Recognition/Lectures2009/SP2009F_Lecture09_Installation%20of%20HTK.pdf

[10] Giampiero Salvi, HTK Tutorial, Dep. of Speech, Music and Hearing, KTH (Royal Institute of Technology), Stockholm, Sweden [Fecha de consulta: 20 Enero 2017] Disponible en: http://www.speech.kth.se/~matsb/speech_rec_course_2003/htk_tutorial.pdf

[11] Nicolas Moreau, HTK Basic Tutorial (HTK), Universite du Quebec, Montreal Canadá [Fecha de consulta: Enero 2017] Disponible en: http://www.labunix.uqam.ca/~boukadoum_m/DIC9315/Notes/Markov/HTK_basic_tutorial.pdf

[12] Ramírez V. G. Diseño un sistema de reconocimiento de voz en Matlab. Universidad San Carlos de Guatemala. [Fecha de consulta :16 de octubre del 2018]. Disponible en : http://biblioteca.usac.edu.gt/tesis/08/08_0223_EO.pdf

[13] Dávila. J. A. B. Un dispositivo para el reconocimiento de caracteres vocálicos para ordenar comandos al televisor. Universidad San Buenaventura de Bógota. [Fecha de consulta: 16 de octubre del 2018]. Disponible en: http://bibliotecadigital.usbcali.edu.co:8080/bitstream/10819/1280/1/Diseno_dispositivo_reconocimiento_Baez_2006.pdf

[14] Muñoz V. C. Desarrollo de un entorno para la interacción multimodal con diferentes aplicaciones en xhtml+voice, Universidad Carlos III de Madrid. [Fecha de consulta: 16 de octubre del 2018]. Disponible en: https://e-archivo.uc3m.es/handle/10016/10632

[15] Goterris F.T. Sistemas de diálogo basados en modelos estocásticos. Universidad Politécnica de Valencia. [Fecha de consulta: 16 de octubre del 2018]. Disponible en: http://www.dsic.upv.es/docs/bib-dig/tesis/etd-11092005-131117/tesisREVISADA6.pdf

[16] Camacho W.A.A. Método de conversión de un diálogo controlado a un discurso en UNLANCE. Universidad Nacional de Colombia. [Fecha de consulta: 16 de octubre del 2018]. Disponible en: http://bdigital.unal.edu.co/10295/1/79692985.2011.pdf

[17] Alvarado G.I.P. Diseño y construcción de un vehículo eléctrico con un variador de velocidad mediante un convertidor de CD-CD. Universidad Tecnológica de Mixteca. [Fecha de consulta: 16 de octubre del 2018]. Disponible en: http://www.itcelaya.edu.mx/ojs/index.php/pistas/article/download/374/362

[18] Rincón J.E. Diseño y construcción de un dispositivo electrónico para la detección de obstáculos como ayuda a personas con discapacidad visual. Universidad de la Salle. [Fecha de consulta: 16 de octubre del 2018]. Disponible en: http://repository.lasalle.edu.co/handle/10185/16456

[19] Valencia J.C.Y. y Mesa M.A.F. PMITO-Plataforma móvil para la investigaciónde técnicas de edometría. Universidad Tecnológica de Pereira. [Fecha de consulta: 16 de octubre del 2018].

[20] Gupta1 S; Jaafar J; Fatimah; W. Bansa A. Feature extraction using mfcc. Universidad tecnológica Petronas. [Fecha de consulta: 16 de octubre del 2018]. Disponible en: http://aircconline.com/sipij/V4N4/4413sipij08.pdf

[21] Ospina E. Hidden markov models (*hmms*)y aplicaciones. Universidad tecnológica de Bolívar. [Fecha de consulta: 20 de octubre del 2018]. Disponible en: http://biblioteca.utb.edu.co/notas/tesis/0049845.pdf

[22] VILLAMIL E. I. H. Aplicaciones en reconocimiento de voz utilizando htk. [Fecha de consulta: 20 de octubre del 2018]. Disponible en: https://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis95.pdf

[23] Grosz,B. Y Sidner, c. Attention intention, and the structure of discourse. Computational Linguistics, No. 12,1986.

[24] GROSZ,B. Discourse analysis. D. Walker(Ed.), Understanding Spoken Lenguage. Ch.IX. ElsevierNorth-Holland, New York pp.235-268.

[25] Abaitua, J., Ruiz, A., Y Zubizarreta, J. Un compilador de LFG y su aplicación al Euskara. Revista Procesamiento del lenguaje natural, No. 9, 1991, pp. 177–191.

[26] Bernsen, N., Dybkjaer, H. Y Dybkjaer, L. Design interactive speech systems, from first ideas to user testing. Springer Verlag, Berlin, 1998.

[27] GATIUS, M. Y GONZÁLEZ, M. Un sistema de diálogo multilingüe dirigido por la semántica. Revista Procesamiento del lenguaje natural, No. 34, 2005.

[28] García, F., Sanchís,E., Hurtado, L. y Segarra, E. Modelos específicos de comprensión en un sistema de diálogo. Revista Procesamiento del lenguaje natural, No. 31, 2003,pp. 99–106.

[29] Alaya T.H.C. Evaluación de la plataforma arduino e implementación de un sistema de control de posición horizontal. Universidad Politécnica Salesiana. [Fecha de consulta: Agosto 2016] Disponible en: https://dspace.ups.edu.ec/bitstream/123456789/5522/1/UPS-GT000511.pdf

[30] S Young, et al. "HTKBOOK", Microsoft Corporation CambriFenanddge University Engineering Department.[Fecha de consulta: Agosto 2016] Disponible en: http://speech.ee.ntu.edu.tw/homework/DSP_HW2-1/htkbook.pdf

[31] P Hwei. ‘HSU-Signals and System’ Chapter 6 Fourier Analysis off Discrete McGraw-Hill, 1997

[32] Ramos F. S. I. Aplicación del programa MATLAB en la resolución de ecuaciones diferenciales aplicado a la materia de Cálculo Tres”. Universidad Católica de Santiago de Guayaquil. [Fecha de consulta: Agosto 2016] Disponible en: http://repositorio.ucsg.edu.ec/bitstream/3317/8531/1/T-UCSG-PRE-TEC-ITEL-202.pdf

10. Anexos

10.1. Anexo 1

SISTEMA DE RECONOCIMIENTO DE VOZ

Este documento tiene como fin ilustrar el diseño de un sistema modelo para el reconocimiento de voz mediante la plataforma de HTK.
El diseño del sistema se puede definir en 10 pasos, los cuales se enumeran a continuación:
Creación de espacio de trabajo.

- Creación de la base de datos.
- Análisis acústico de las señales.
- Definición de los Modelos ocultos de Márkov (HMM).
- Entrenamiento de los HMM's.
- Reestimación de los modelos.
- Modelos gramaticales.
- Construcción de la red de trabajo.
- Reconocimiento
- Prueba de rendimiento

Sobre el reconocimiento de voz

El reconocimiento de voz consiste básicamente en la comunicación hombre maquina, a partir de la detección de extremos (inicio y fin de la palabra) o la detección del segmento, con el fin de entregar al sistema la palabra reconocida.
Para poder entregar la palabra reconocida, el segmento tiene que pasar por una debida estimación, seguido a esto se deben someter cada uno de los modelos a una reestimación hasta que converjan, es decir, que los modelos en cada estimación reducen la variación de su transición de estados. A este proceso se conoce como digitalización de la señal, proceso que permite obtener los elementos de medición, los cuales permiten conocer su comportamiento.
Este método ha sido muy explotado en el siglo XXI debido a su utilidad, ya que elimina engorrosas tareas como el tener que escribir textos, sistemas de control, entre otros.

1. Creación de espacio de trabajo.

Para el diseño de un sistema de reconocimiento de voz es necesario un espacio de trabajo, este debe estar contenido en la carpeta HTK, donde su función principal es contener todos los datos del diseño, tales como señales, MFCC, entre otros. Para tener mejor orden en el espacio de trabajo se sugiere crear una carpeta para cada proyecto, debido que cada proyecto contiene diferentes modelos, señales entre otros archivos.

Esta carpeta debe contener el siguiente directorio:

- **data:** Esta carpeta contiene una carpeta llamada mfcc, la cual contiene los coeficientes mfcc de todas las señales.
- **lab:** Carpeta que contiene los archivos .lab , en los cuales se etiquetaron los intervalos sonoros y no sonoros.
- **model:** Carpeta que contiene los datos entrenados por el sistema.
- **sig:** Carpeta que contiene las señales a reconocer.

2. Base de datos.

Al diseñar un programa de reconocimiento de voz es necesario crear una base de datos ya que los modelos deben ser entrenados con datos existentes en este caso señales grabadas.

2.1. Señales Grabadas.

La base de datos son audios los cuales se categorizan en las clases a reconocer en el sistema. Anteriormente se mencionaba las dos clases que se trabajaran en el sistema modelo Si & No, por consiguiente, se debe de realizar grabaciones en las cuales se pronuncie 30 veces sí y 30 no, con duración aproximadamente de 5s.
Las grabaciones se realizaron mediante el programa de Audacity, ver anexo 2. Realizadas las 60 grabaciones con una frecuencia 16KHz con extensión .wav se parte a los cálculos de los intervalos. **Nota:** Para evitar problemas con el desarrollo del sistema nombrar carpetas y archivos con minúscula.
NOTA: A PARTIR DE VARIAS PRUEBAS SE DETERMINO QUE PARA LA GRABACIONES DEBEN SER COMO MINIMO DE 4 A 5 SEGUNDOS, ADEMAS DE QUE SU PARAMETRIZACION NO DEBE SER TAN ROBUSTA Y EXACTA YA QUE ESTO PUEDE CAUSAR QUE NO SE RECONOZCAN LA PALABRAS, NI SUS TRAMAS DE SILENCIO.

2.2. Parametrización y etiquetado de las señales.

La parametrización consiste en conocer el tiempo de inicio y fin de cada palabra dentro del fichero de audio. Al identificar los intervalos sonoros y no sonoros es necesario tener en cuenta el etiquetado de los audios puesto que es indispensable para la base de datos categorizar los intervalos de las señales. Los archivos en los cuales se van a escribir estas etiquetas son ficheros de texto con extensión .lab .
Los archivos de etiquetas y los respectivos intervalos sonoros y no sonoros con tendrán un aspecto como se muestra a continuación:

0 5165000 sil
5165000 7171875 no
7171875 18591250 sil

Cada una de estas líneas representa el tiempo inicial, el tiempo final y la etiqueta de la clase a la que pertenecen esas tramas temporales. Por ejemplo, en el anterior ejemplo, la primera trama de silencio (sil) empieza en el instante cero (0) y finaliza en el instante 5165000.

Nótese que las marcas de tiempo están dadas en centenas de nano segundo, es decir, el intervalo de tiempo del silencio inicial va del segundo 0 al segundo 0,5165. Dicho de otra forma, para representar este valor en centenas de nanosegundos se hace necesario multiplicarlo por 10^7

Una forma de obtener los valores temporales de estas marcas es empleando el software Audacity. Este software ofrece la posibilidad de ver el tiempo de cada evento. El inconveniente que tiene es que, al mostrar el tiempo, ofrece pocas cifras decimales y esto hace perder precisión temporal al etiquetar las muestras. Por esta razón es recomendable ver la duración de los eventos en términos de muestras (en vez de tiempo). Por ejemplo, el evento que aparece en la figura 1 ocurre en la muestra 50290.

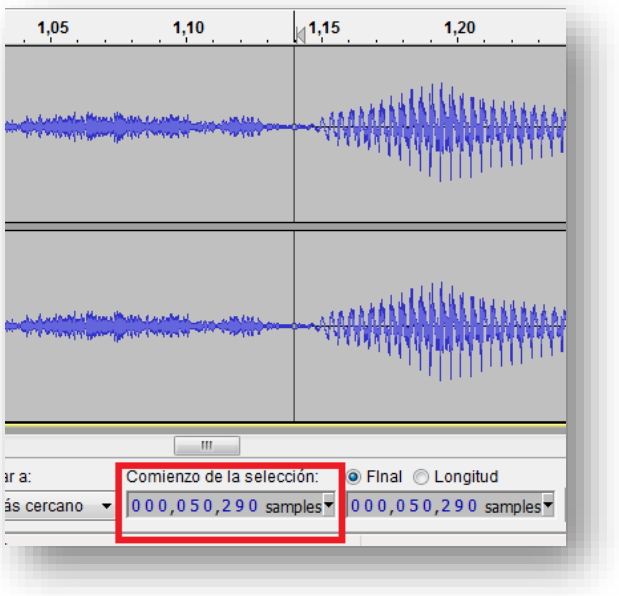

Figura 1. Ventana de tiempo en el software 2 Audacity.

Una vez tengamos el valor de la muestra en que ocurren los eventos, es posible, mediante una regla de tres simple, obtener su equivalente en el tiempo. Por ejemplo: si se está empleando una frecuencia de muestreo de 16kHz, el cálculo es el siguiente

$$t = \frac{50290}{16000} = 3{,}143125\ s$$

Una vez obtenido este valor se multiplica por 10^7 para tenerlo en centenas de nanosegundos.

Se pueden ver más detalles acerca de la operación del software Audacity en el Anexo 2 (3. Menú desplegable del ventaneo).

NOTA: solo se un carácter deja un espacio entre los límites y entre el intervalo y la etiqueta.

3. Análisis acústico de las señales.

El análisis acústico consiste en la conversión de una señal de onda a un vector de coeficientes cepstrales en la frecuencia de MEL (MFCC). Los MFCC son coeficientes para la representación del habla basados en la percepción auditiva [2].

Para la obtención de los coeficientes MFCC es necesario realizar los siguientes pasos:
Pre-énfasis (PE).
Ventaneo o muestreo (W).
Transformada de Fourier (FFT).
Conversión a escala logarítmica (LOG).
Transformada discreta del coseno (DCT).
En el diagrama de bloques de la figura 2 se puede observar el orden de análisis que se realiza.

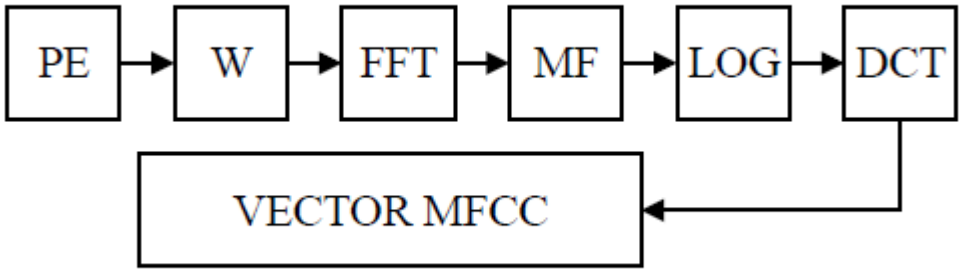

Figura 2. Esquema de obtención de los MFCC's [2].

Configuración de parámetros para el Análisis acústico.
La configuración de parámetros se realiza en un script que debe llevar como nombre analysis.conf .
El script va a describir la configuración del análisis acústico como se muestra en la figura 3.

```
# Configuracion para analisis acustico
#
SOURCEFORMAT = WAV          # Define el tipo de formato del archivo de entrada
TARGETKIND = MFCC_0_D_A # Identifica los coeficientes que se van a usar
                            # el 0 identifica al coef c0
                            # la D identifca los coeficientes DELTA (es decir, las derivadas
                            # de [c0,c1,...,c12]
                            # la A identifica los coeficientes de ACELERACION (es decir,
                            # las segundas derivadas de [c0,c1,...,c12].
                            # la Z hace normalización cepstral CMN
                            # En total son 39 parámetros
# Unidad = 0.1 microsegundos (las mismas unidades en centenas de nanosegundos)
WINDOWSIZE = 250000.0       # 25 ms - ancho de una ventana de analisis
TARGETRATE = 100000.0       # 10 ms - periodicidad de la ventana

NUMCEPS = 12                # Numero de coefs cepstrales (van de c1 a c12)
USEHAMMING = T              # TRUE para el uso de ventana de Hamming
PREEMCOEF = 0.97            # Coeficiente para filtrado de pre-enfasis
NUMCHANS = 26               # Numero de canales de bancos de filtros
CEPLIFTER = 22              # Longitud del filtrado cepstral

# El fin
```

Figura 3. Configuración de análisis acústico.

3.2. Configuración de locación.

La configuración de locación es un script con extensión .txt el cual debe por nombre targetlist.txt .Este script contiene las direcciones y formato de las señales a las cuales se va realizar el análisis acústico. Seguido a esto se especifica el lugar en el cual se va a guardar el análisis acústico de cada señal junto al formato resultante.

Este script debe estar incluido en el espacio de trabajo, debido a que él cuenta con un direccionamiento relativo, por lo cual no es necesario especificar su ubicación desde disco raíz solo la carpeta en que se encuentra y su formato. El script debe tener el siguiente aspecto:

sig/No/No_01.wav data/mfcc/No_01.mfcc

Creados los dos scripts donde se describen las configuraciones acústicas, las ubicaciones de las señales, se procede a ejecutar el comando de HTK puesto que al compilar dará como resultado los coeficientes cepstrales de cada señal. Este comando se ejecuta desde nueva instancia de consola como se muestra en la figura 4. Nota: Se sugiere que la dirección de la nueva instancia se encuentre direccionada al espacio de trabajo de HTK, puesto que los comandos cuentan con un direccionamiento relativo.

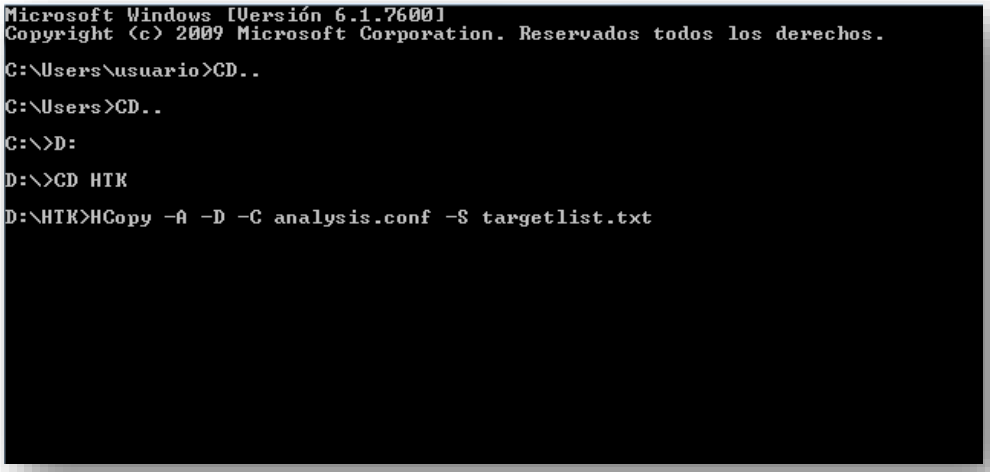

Figura 4. Compilación desde el espacio de trabajo del comando HCopy –A –D –C analysis.conf –S targetlist.txt .

4. Definición de los Modelo oculto de Márkov.

Los modelos ocultos de Márkov (HMM) constituyen una herramienta de modelamiento altamente flexible, inicialmente utilizada en el campo del reconocimiento automático del habla, que ha encontrado en los últimos años numerosas aplicaciones en áreas científico-técnicas muy diversas [3].

Para este caso se tratan 3 eventos acústicos: si, no silencio. Cada uno cuenta con un modelo oculto de Márkov, el cual tiene una topología determinada por:

- Numero de estados.
- Forma de funciones (Asociado a cada estado).
- Disposición de las transiciones de estados.

En este caso la topología del modelo acústico cuenta con 6 estados, 4 son activos (S2, S3, S4, S5) y dos son estados no emisores (S1, S6), esto quiere decir sin función de observación siendo de uso exclusivo de HTK. En la figura 5 se muestra la topología del sistema, identificando los estados, las posibles transiciones (ai) de estados y las distribuciones gaussianas (bj) [1].

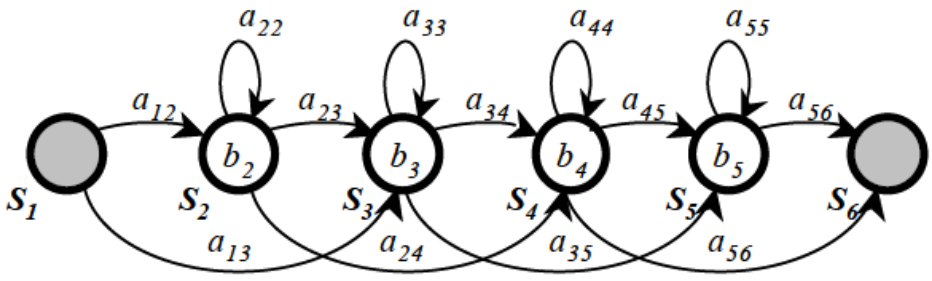

Figura 5. Topología del sistema [1].

El modelo de cada palabra se debe inicializar en un script. Script el cual cuente con extensión .txt , llevando por nombre hmm_si, hmm_no, y hmm_sil. Los scripts creados tienen como función inicializar los parámetros como valores estimados de las palabras. Estos scripts deben estar contenidos en el directorio **model/proto/.** El script de tener el aspecto como el que se ilustra en la figura 6.

```
~o
<STREAMINFO> 1 39
<VECSIZE> 39<NULLD><MFCC_D_A_0><DIAGC> // DIMENSION DEL VECTOR Y EL TIPO DE COEFICIENTES A OBTENER
~h "si"                    // DESCRIPCION DE UN HMM llamado " sí" .
<BEGINHMM>
<NumStates> 6             //NUMERO DE ESTADOS
<State> 2                 //ESTADO 2
<Mean> 39                //DA EL VECTOR MEDIO(EN UN ESPACIO DE OBSERVACION CON UNA DIMENSION DE 39)DE LA OBSERVACION ACTUAL
función
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39               //VARIANZA DEL ESTADO 2
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 3                  //ESTADO 3
<Mean> 39
0.0 0.0 (...) 0.0
<Variance> 39              //VARIANZA DEL ESTADO 3
1.0 1.0 (...) 1.0
<State> 4                 //ESTADO 4
<Mean> 39
0.0 0.0 (...) 0.0
<Variance> 39            //VARIANZA DEL ESTADO 4
1.0 1.0 (...) 1.0
<State> 5               //ESTADO 5
<Mean> 39
0.0 0.0 (...) 0.0
<Variance> 39          //VARIANZA DEL ESTADO 5
1.0 1.0 (...) 1.0
<TransP> 6                //ES LA PROBABILIDAD DE TRANSICIÓN DEL ESTADO I AL ESTADO J
0.0 0.5 0.5 0.0 0.0 0.0  //LOS VALORES NULOS INDICAN QUE LA TRANSICIONES CORRESPONDIENTES NO ESTÁN PERMITIDOS.
0.0 0.4 0.3 0.3 0.0 0.0
0.0 0.0 0.4 0.3 0.3 0.0
0.0 0.0 0.0 0.4 0.3 0.3
0.0 0.0 0.0 0.0 0.5 0.5
0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figura 6. Plantilla modelo para la inicialización de modelos.

NOTA: Se anexa el archivo modelo para los modelos ocultos de markov “hmm_name” donde simplemente es necesario cambiar el nombre del modelo, el cual se señala con un ovalo rojo en la figura 12, este archivo no cuenta con ningún tipo extensión.

Es necesario que antes iniciar la compilación de los comandos se conozca los archivos con extensión .bat puesto que serán una herramienta muy importante en el proceso de la compilación debido a que reducirán el tiempo de realización.

Son archivos de texto sin formato, guardados con la extensión .bat que contienen un conjunto de instrucciones MS-DOS. Lo que significa que al ser ejecutado este archivo las órdenes contenidas son ejecutadas en grupo, de forma secuencial permitiendo automatizar diversas tareas, en este caso la ejecución de diferentes comandos puesto que algunos se tienen que compilar múltiples veces.

Estos archivos se crean de la siguiente manera:

- Crear un fichero con extensión .txt (presionar el botón derecho del mouse – Nuevo – Documento de texto).
- Cambiar la extensión .txt por .bat

Si se desea modificar el ejecutable que contiene el fichero .bat , se debe abrir con Notepad++ .

5. Entrenamiento de los HMM'S.

Los parámetros de los modelos ocultos de Márkov deben estar debidamente inicializados con datos de entrenamiento, a fin de permitir una convergencia rápida y precisa del algoritmo de entrenamiento. HTK ofrece 2 herramientas de inicialización diferentes: Hinit y HCompv [1].

5.1.HInit

Comando cuya función es inicializar los modelos (si, no y sil[silencio]), modificando los valores estimados con los cuales se inicializo por valores entrenados.
La estructura de comando HInit se puede observar en la figura 7.

```
HInit -A -D -T 1 -S trainlist.txt -M model/hmm0
 -H model/proto/hmmfile -l label -L Label_dir nameofhmm
```

Figura 7. Comando HInit [1].

La estructura del comando HInit cuenta con parámetros tales como:
trainlist.txt: Es el documento que contiene la(s) dirección(es) donde se encuentra(n) los coeficientes cepstrales de la(s) señal(es)[MFCC] que se va a entrenar .

- **model/hmm0:** Es el nombre del directorio que debe ser creado antes de la compilación del comando. Dando como resultado la inicialización del HMM con valores de entrenamiento.
- **model/proto/:** Dirección en la cual se encuentra el modelo a inicializar.
- **hmmfile:** Es el archivo que contiene la descripción del prototipo de los HMM, para este caso hmm_no, hmm_si, y hmm_sil.
- **label:** Indica la etiqueta que se usa dentro de los datos de entrenamiento (lab).
- **label-dir:** Dirección donde se encuentra los archivos de etiqueta.
- **nameofhmm:** Nombre del modelo que se inicializo.

5.2.HCompv

Comando cuya función es la normalización de los datos a partir del vector de varianza global multiplicado por un factor que se fija en el comando. La estructura de comando HCompv se puede observar en la figura 8.

```
HCompv -A -D -T 1 -S trainlist.txt -M model/hmm0flat
-H model/proto/hmmfile -f 0.01 nameofhmm
```

Figura 8.Comando Hinit.

En la estructura se logran reconocer parámetros similares al del comando HInit los cuales cuentan con la misma función, sin embargo se encuentran parámetros nuevos tales como:

model / hmm0flat: Directorio de salida, el cual debe ser diferente a la utilizada formalmente por el comando HInit.
-f 0.01: Factor que multiplica la varianza global.

Como resultado a la ejecución del comando HCompv se obtendrá como salida los varianza de cada señal y un archivo llamado varFloor1 o mejor conocido como " varianza bajo macro " el cual va a contener un vector que define la varianza global de todas las señales.

6. Reestimación de los modelos.

La reestimación de los modelos consiste en volver a determinar los parámetros calculados por el comando HInit a partir de la "la varianza bajo macro" obtenida por comando HCompv. Este proceso se realiza para obtener los valores óptimos para probabilidades de transición, más media y vectores de la varianza de cada función de observación. Para la reestimación se tiene el comando HRest cuya estructura es la de figura 9.

```
HRest -A -D -T 1 -S trainlist.txt -M model/hmmi -H vFloors -H
model/hmmi-1/hmmfile -l label -L labdel_dir nameofhmm
```

Figura 9. Estructura del comando HRest.

Varios de los parámetros que se encuentran en la estructura del comando HRest ya son conocidos por HInit y HCompv, pero también se encuentran otros no conocidos, que se darán a conocer a continuación:

- **Model/hmmi:** Directorio de salida para los modelos. Donde "i" es el subíndice que indica la iteración en la que se encuentra.
- **vFloors:** Este parámetro es "la varianza bajo macro" de los modelos, el cual se tiene que modificar y esto se debe a que al realizar las pruebas pertinentes se encontró necesario añadir tres líneas de texto como las que se señalan en la figura 10.

```
~v varFloor1
<Variance> 39
 2.787563e-01 1.247354e-01 1.171571e-01
 1.493054e-01 1.829179e-01 2.153076e-01
 1.824232e-01 1.927942e-01 1.827625e-01
 1.633700e-01 1.497350e-01 1.406506e-01
 8.114429e-01 6.102916e-03 5.852143e-03
 6.692554e-03 9.074634e-03 1.108821e-02
 1.223366e-02 1.376805e-02 1.482704e-02
 1.549832e-02 1.450212e-02 1.356561e-02
 1.227011e-02 8.409141e-03 8.698062e-04
 9.482017e-04 1.194519e-03 1.659967e-03
 2.048119e-03 2.375631e-03 2.682189e-03
 2.869322e-03 3.042837e-03 2.888390e-03
 2.733949e-03 2.439121e-03 1.243732e-03
```

```
1 ~o
2 <VECSIZE> 39
3 <MFCC_0_D_A_Z>
4 ~v varFloor1
5 <Variance> 39
6  2.659268e-01 1.200291e-01 1.226369e-01
   1.470984e-01 1.830874e-01 2.087722e-01
   1.790623e-01 1.918293e-01 1.797172e-01
   1.641790e-01 1.479923e-01 1.388642e-01
   7.691528e-01 6.019037e-03 5.660357e-03
   6.739932e-03 8.979498e-03 1.109636e-02
   1.224312e-02 1.366979e-02 1.476371e-02
   1.537822e-02 1.456194e-02 1.357242e-02
   1.224951e-02 7.912585e-03 8.533819e-04
   9.214656e-04 1.199756e-03 1.648857e-03
   2.046733e-03 2.374895e-03 2.667909e-03
   2.862020e-03 3.029535e-03 2.899668e-03
   2.740344e-03 2.436723e-03 1.168170e-03
```

Figura 10. Modificación al vFloors.

- **model/hmmi-1/hmmfile:** directorio en el cual se encuentran los datos a entrenar. Donde "hmmi-1" es el directorio anterior, de manera que va a reestimar datos ya entrenado, con el fin de tener valores óptimos.

Nota: Antes de la ejecución del comando los directorios hmmi se tienen crear dentro de la carpeta model que se encuentra en el espacio de trabajo. El número de capetas está definido por "i" que es el número de iteraciones que se realizaran (4 iteración permite tener valores óptimos para la transición), por tanto, el número de iteraciones que realice va a ser el número de carpetas que se tiene que crear.

7. Modelos Gramaticales.

La gramática de los modelos está dado por un fichero de texto, el cual debe llevar por nombre gram.txt . Este fichero de texto define las probabilidades para ser pronunciadas las palabras propuestas a reconocer como se ilustra en la figura 11.

```
$WORD= si|no;  /* Palabra(s) a reconocer */
({START_SIL}[$WORD]{END_SIL}) /* Estructura de la gramtica*/
```

Figura 11. Estructura del fichero gram.txt .

Al ser definida la estructura es necesario determinar un diccionario por consiguiente se crea un fichero llamado dict.txt , siendo este el que contiene las palabras que se reconocen, la etiqueta con la cual se reconoce y la salida al ser reconocida como muestra en la figura 12.

```
si [si] si
no [no] no
START_SIL [sil] sil
END_SIL [sil] sil
```

Figura 12. Estructura del fichero dict.txt .

Es de aclarar que los elementos que se encuentren en la parte izquierda se refieren a la variable del modelo gramatical, los elementos que encuentren entre los corchetes son la palabra que emite al reconocedor, y los elemento que se encuentran a la derecha se refiere a las etiquetas utilizada en lo HMM como señala en la figura 13.

```
~o
<STREAMINFO> 1 39
<VECSIZE> 39<NULLD><MFCC_D_A_0><DIAGC>
~h "no"
<BEGINHMM>
<NUMSTATES> 6
<STATE> 2
<MEAN> 39
 -1.894089e+001 4.685268e+000 6.228234e-001 5.112936e+000 1.417273e+000 2.866990e+000 1.026350e+000 2.707728e+000
 -1.954210e+000 3.450622e+000 3.434633e-002 -3.110160e-001 5.101637e+001 3.839724e-002 -4.960793e-002 -2.160625e-002
 -8.227041e-003 -8.220736e-002 -3.705471e-002 -4.137742e-002 -8.269224e-002 1.535077e-002 6.040480e-002 -1.945469e-002
 3.367605e-003 4.565461e-002 -1.120936e-002 -4.123836e-003 -9.371184e-003 -9.810747e-003 2.572888e-002 1.914810e-002
 -7.447585e-003 2.667885e-002 2.601345e-002 -1.328061e-002 -1.286554e-002 -9.953060e-003 -8.828082e-003
```

Figura 13. Etiqueta 'no' del HMM.

8. Construcción de la red de trabajo.

La construcción de las redes de trabajo consta de la conexión de los posibles estados, con base a la estructura gramatical. Siendo la estructura gramatical la que determina la secuencia de estados.
La red de trabajo se crea a partir del comando HParse cuya estructura se muestra en la figura 14.

```
HParse -A -D -T 1 gram.txt net.slf
```

Figura 14. Estructura del comando HParse.

Comando el cual contiene dos parámetros:

- **gram.txt:** Es el fichero que contiene la estructura gramatical.
- **net.slf:** Es el fichero en el cual se va obtener la red de trabajo.

Para corroborar que se creó una estructura de trabajo adecuada se emplea el comando HSGen cuya estructura se ilustra en la figura 15.

```
HSGen -A -D -n 10 -s net.slf dict.txt
```

Figura 15. Estructura del comando HSGen.

Comando el cual contiene dos parámetros:

- **net.slf:** Es el fichero en el cual se obtuvo la red de trabajo.
- **dict.txt:** Fichero el cual contiene el diccionario (creado en el paso 7).

9. Reconocimiento

Este procedimiento se puede realizar de dos maneras, reconocimiento de voz archivo o reconocimiento de voz por directo.
Estos sistemas cuentan con una gran diferencia y es que en el reconocimiento de voz por archivo estos contienen todo el comportamiento y se tiene en cuenta la varianza global para el óptimo reconocimiento mientras que en el reconocimiento por voz no se conoce su comportamiento, en consecuencia, no se tiene en cuenta la varianza global influyendo en la eficiencia del sistema.

9.1.Reconocimiento de voz por archivo.

Para el reconocimiento de voz por archivo es necesario ejecutar el comando HVite, el cual puede reconocer un archivo o una lista archivos.

9.1.1. Reconocimiento de un archivo

Para el reconocimiento de un único archivo se ejecuta un el comando que se muestra en la figura 16.

```
HVite -A -D -T 1 -H hmmsdef.mmf -i reco.mlf -w net.slf
dict.txt hmmlist.txt input.mfcc
```

Figura 16. Comando para el reconocimiento de un archivo.

Este comando reconoce los modelos que se le ingresan por lo tanto es necesario tener en cuenta los siguientes parámetros:

- **reco.mlf:** Este fichero debe ser creado puesto que contiene la transcripción de los intervalos en los que se reconoce la palabra que se mencionó se ilustra en figura 17. Al ser ejecutado el comando, se realizará una transcripción, siendo una hipótesis de la palabra mencionada, como se ilustra en la figura 18.

```
#!MLF!#
"data/mfcc/no_h01_01.rec"
sil
no
sil
```

Figura 17. Contenido del archivo **reco.mlf** antes de la compilación del comando.

```
#!MLF!#
"data/mfcc/no_h01_01.rec"
0 4200000 sil -2331.602295
4200000 5100000 sil -708.960693
5100000 7300000 no -1761.585815
7300000 18500000 sil -5877.106445
```

Figura 18. Contenido del archivo reco.mlf después de la compilación del comando.

- **hmmlist.txt:** Este fichero debe ser creado debido a que contiene la lista de las etiquetas utilizadas en los modelos (HMM), en este caso [si, no y sil].

- **input.mfcc:** Es el dato que se ingresa a reconocer.

- **hmmsdef.mmf:** Este fichero está compuesto por las definiciones de los HMM.
  El fichero tiene dos formas de ser definido, para un solo modelo (HMM) o para múltiples modelos (HMM).
  En el caso de un solo modelo es necesario definir la ruta de donde encuentra el modelo a reconocer, mientras que para múltiples modelos es necesario reunir el contenido de las definiciones (hmm_no, hmm_si, hmm_sil) en un archivo de texto definido como hmmsdef.mmf.
  Los modelos contenidos en hmmsdef.mmf pertenecen a la ultima carpeta que se entrenó (model\hmm4) de los HMM, donde su estructura se definiría como se ilustra en la figura 19.

```
~o
<STREAMINFO> 1 39
<VECSIZE> 39<NULLD><MFCC_D_A_0><DIAGC>
~h"si"
<BeginHMM>
        (definicion.....)
<EndHMM>
~h"no"
<BeginHMM>
        (definicion.....)
<EndHMM>
~h"sil"
<BeginHMM>
        (definicion.....)
<EndHMM>
```

Figura 19. Estructura del archivo hmmsdef.mmf.

9.1.2. Reconocimiento de múltiples archivos

Para el reconocimiento de una lista de archivos se ejecuta el comando que se muestra en la figura 20.

```
HVite -A -D -T 1 -S testlist.txt -H hmmsdef.mmf
-i rec.mlf -w net.slf dict.txt hmmlist.txt
```

Figura 20. Comando para el reconocimiento múltiples archivos.

Este comando cuenta con parámetros similares al comando anterior, donde su contenido es el mismo, sin embargo, cuenta con otros parámetros como:

- **testlist.txt:** Este archivo contiene una lista con las direcciones de los modelos con los cuales se realizará la prueba de rendimiento(data/mfcc/name.mfcc).
- **rec.mlf:** Este parámetros es similar a reco.mlf pero con un distintivo y es que no contiene una hipótesis, este contiene todas las direcciones y etiquetas de los modelos que se les desea hacer la prueba de rendimiento como se muestra en la figura 21.

```
#!MLF!#
"data/mfcc/no_h01_01.rec"
sil
no
sil
"data/mfcc/no_h01_02.rec"
sil
no
sil
"data/mfcc/no_h01_03.rec"
sil
no
sil
"data/mfcc/no_h01_06.rec"
sil
no
sil
```

Figura 21. Contenido del archivo reco.mlf.

9.2.Reconocimiento de voz por directo.

El proceso del reconocimiento de voz por directo consta de la ejecución del comando HVite pero con diferente estructura como se ilustra en la figura 22.

```
HVite -A -D -T 1 -C directin.conf -g -H
hmmsdef.mmf -w net.slf dict.txt hmmlist.txt
```

Figura 22. Estructura del comando HVite para el reconocimiento de voz por directo.

Algunos parámetros ya están definidos como lo son: hmmlist.txt, hmmsdef.mmf, dict.txt y net.slf. Sin embargo, este comando cuenta con un parámetro nuevo, este debe ser creado con el nombre **directin.conf**, cuyo fin es permitir la extracción directa de los coeficientes acústicos de la señal de entrada, este archivo debe contener unos parámetros de configuración análisis acústica los cuales se ilustran en la figura 23.

```
# HVite configuracion de variables para la entrada de audio directo
#
#Parametros de la señal de entrada
SOURCERATE = 625.0   # = 16KHz
SOURCEKIND = HAUDIO
SOURCEFORMAT = HTK
#Conversionte parametros de la señal de entrada
TARGETKIND = MFCC_0_D_A    #Identifica los coeficientes que se utilizaran
WINDOWSIZE = 250000.0      # = 25ms = longitud de tiempo de cada ventana
TARGETRATE = 100000.0      # = 10ms = periodicidad de la ventana
NUMCEPS = 12               # Numero de coeficientes MFCC(Desde C1 hasta C12)
USEHAMMING = T             #El uso de la funcion de hamming para marcos de ventanas
PREEMCOEF = 0.97           # Pre-enfasis de los coeficientes
NUMCHANS = 26              # Numeros de "filterbank" canales
CEPLIFTER = 22             # Duracion de los ceptrales

# Define la señal  para usar como un control remoto
AUDIOSIG= -1               #EVALUACION NEGATIVA = CONTROL DE PULSACION

# THE END
```

Figura 23. Parámetros de configuración análisis acústica.

10. Prueba de rendimiento

Para un sistema de reconocimiento de voz es fundamental conocer su rendimiento puesto que allí se concluye si el sistema es eficiente o deficiente el reconocimiento.
Este proceso se realiza mediante la evaluación del sistema con archivos modelados mas no entrenados en el sistema. Al decir modelado se refiere a realizar el mismo proceso de los pasos 2 y 3, a los archivos con los cuales se evaluará el sistema.
Al realizar la prueba de rendimiento es fundamental crear el siguiente archivo:

ref.mlf: Este fichero está compuesto por los datos de los archivos con los cuales se evaluará el sistema (dirección y etiqueta con las que se parametrizaron), como se ilustra en la figura 24.

```
#!MLF!#
"Lab/no_h01_23.lab"
sil
no
sil
.
"Lab/no_h01_12.lab"
sil
no
sil
.
"Lab/no_h01_20.lab"
sil
no
sil
.
```

Figura 24. Estructura del fichero ref.mlf.

labellist.txt: Archivo que contiene las etiquetas que se reconocerán, para este caso si, no y sil las cual deben estar separadas por un salto de línea.

Al contar con el archivo ref.mlf y el labellist.txt, es fundamental copiar los ficheros .lab de los archivos con los cuales se evaluara el sistema en la carpeta que contiene los MFCC (data/mfcc). Seguidos los pasos anteriores se compila el comando de la figura 25.

```
HResults -A -D -T 1 -e ??? sil -I ref.mlf labellist.txt
rec.mlf > results.txt
```

Figura 25. Comando para el cálculo de rendimiento.

Comando cuya función es realizar la prueba de rendimiento mediante la lista que se creó a partir de las hipótesis del sistema, donde es de aclarar que en el código se coloca –e ??? sil con el fin de que al momento de comparar no tenga en cuenta los silencio y >results.txt es el archivo que se destina para que guarde la respuesta del sistema como se muestra en la figura 26.

```
====================== HTK Results Analysis =======================
  Date: Tue Dec 03 19:12:58 2002
  Ref : .\ref.mlf
  Rec : .\rec.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=80.00 [H=8, S=2, N=10]
WORD: %Corr=80.00, Acc=80.00 [H=8, D=0, S=2, I=0, N=10]
====================================================================
```

Figura 26. Archivo results.txt.

BIBLIOGRAFIAS ANEXOS

[1]. N.Moreau, HTK: Basic tutorial.

[2]G.A.M. Mascorro y G.A.Torres, Sistema para identificación de hablantes robusto a cambios en la voz.

[3]. F. Rodríguez, S. Bautista, Modelos ocultos de Markov para el análisis de patrones espaciales.